Titre: Title:	Méthodes d'inspection automatique d'infrastructure par robot mobile
Auteur: Author:	André Phu-Van Nguyen
Date:	2017
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Nguyen, A. P.-V. (2017). Méthodes d'inspection automatique d'infrastructure par robot mobile [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2935/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2935/
Directeurs de recherche: Advisors:	Jérôme Le Ny, & David Saussié
Programme: Program:	génie électrique

UNIVERSITÉ DE MONTRÉAL

MÉTHODES D'INSPECTION AUTOMATIQUE D'INFRASTRUCTURE PAR ROBOT MOBILE

ANDRÉ PHU-VAN NGUYEN
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

MÉTHODES D'INSPECTION AUTOMATIQUE D'INFRASTRUCTURE PAR ROBOT MOBILE

présenté par : NGUYEN André Phu-Van
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GOURDEAU Richard, Ph. D., président
M. LE NY Jérôme, Ph. D., membre et directeur de recherche
M. SAUSSIÉ David, Ph. D., membre et codirecteur de recherche
M. PAULL Liam, Ph. D., membre

DÉDICACE

Je dédie ce travail à la mémoire de Dopey,
ami, frère, chien,
woof woof.

REMERCIEMENTS

Je tiens d'abord à remercier mes amis et collègues de laboratoire Alexandre Borowczyk, Dang Quang Nguyen, Duc Tien Nguyen et Gabriel Guilbert pour les fructueuses discussions théoriques et surtout pour avoir participé avec moi au DJI Challenge 2016. Ce fut la compétition d'ingénierie la plus difficile à laquelle j'ai participé durant mon cheminement académique et ce fut un honneur de l'avoir fait avec vous.

Je remercie aussi mes collègues Olivier Gougeon, Justin Cano, Jérémie Pilon, Catherine Massé et Juliette Tibayrenc pour les bons moments passés dans le Laboratoire de Robotique Mobile et des Systèmes Autonomes.

Je remercie Laurence Lebel, Alexandre Marceau-Gozsy, Simon Dufour, Mélanie Harvey, Simon Bourgault-Côté, Yoann Arpin, Marie Tardif-Drolet, Gabriel Brassard, Khaldoun El-Hajj, Marc Castanet, Alessandro Scola, Hubert Courteau-Godmaire, Félix Amyot et toutes les autres personnes que j'ai pu côtoyer lors de la construction de la voiture solaire Esteban VI. Les deux années que j'ai passé dans l'équipe avec vous étaient essentielles à ma formation d'ingénieur et m'ont amenées à étendre mes connaissances bien au-delà des sujets de l'informatique.

Je remercie Antoine Mignon, Marc-André Ruel, David Thibodeau, Laurier Loiselle, Antonio Sanniravong, Constant Rietsch, Jean-Aleandre Barszcz, Quentin Gili, Alexandre St-Onge, David Binet et toutes les autres personnes qui on aidé de près ou de loin à la formation et l'opération de l'équipe du multirotor autonome Élikos. Je suis fier d'avoir bâti avec vous un projet d'envergure dont la pérenité est maintenant assurée par une fortre tradition d'excellence. Mes deux ans en tant que directeur du projet (et nos deux premières place à l'*International Aerial Robotics Competition*) ont été un facteur décisif dans ma poursuite du sujet de la robotique aérienne aux cycles supérieurs.

Finalement j'aimerais remercier mes directeurs de recherche Jérôme Le Ny et David Saussié pour avoir accepté de superviser mes travaux de maîtrise et pour m'avoir aidé sans relâche à travers tout mon parcours de maîtrise.

RÉSUMÉ

La robotique est un domaine voué à la création de machines permettant d'aider ou de remplacer les humains lors de tâches difficiles ou dangereuses. Plus souvent utilisés dans le secteur manufacturier ou la recherche et sauvetage, les robots trouvent maintenant leur place dans les secteurs de l'arpentage et l'inspection d'infrastructure civile grâce à leur mobilité et leur capacité à effectuer des tâches répétitives. Cependant, ces robots sont souvent encore opérés à distance et ne possèdent que de l'autonomie partielle, étant toujours incapable de prendre des décisions eux-mêmes.

Dans ce mémoire, nous tentons de répondre à ce problème en présentant deux méthodes permettant de guider un robot autour d'une structure pour en faire l'inspection. Premièrement, nous attaquons le problème d'inspection au sol de structures fermées. Fonctionnant au moyen d'un robot terrestre équipé d'une caméra de profondeur, notre méthode permet de cartographier la surface visible d'une structure fermée. En particulier, nous proposons un système qui cherche à effectuer l'inspection tout en minimisant la possibilité d'erreur sur le système de localisation du robot. Deuxièmement, nous proposons un système spécifiquement pour l'inspection d'éoliennes. Au moyen d'un véhicule aérien non habité et de scanneurs lasers 2D, notre méthode permet à un quadricoptère de décoller du sol et suivre la tour pour ensuite automatiquement parcourir la surface avant des pales d'une éolienne.

Nos systèmes sont tous les deux implémentés et démontrés dans des environnements de simulation. L'inspection terrestre est démontrée sur un vrai robot dans un scénario d'inspection intérieur où nous démontrons aussi comment traiter le bruit non pris en compte dans la simulation. Finalement, nous présentons des résultats préliminaires sur le déploiement de l'inspection aérienne.

ABSTRACT

Robotics is a branch of engineering dealing with the creation of machines capable of helping or even replacing humans in tasks which can be dull, dirty or dangerous. Most commonly used in the fields of manufacturing and search and rescue, in recent years robotics has found its place in surveying and infrastructure inspection thanks to their increased mobility and their ability to tackle repetitive tasks. However, today's robots are still often remote controlled with only partial autonomy and still aren't capable of decision making and navigation.

In this thesis, we address this problem by presenting two methods allowing a robot to navigate around a structure to inspect its surface. First, we tackle the problem of ground based inspections of the visible part of a bounded closed structure. Using an unmanned ground vehicle equipped with a depth sensor, our method allows us to map the entire surface of the structure. Our specific contribution here, is a system capable of performing the inspection while also minimizing errors on the robot's localization system. Also known as "active slam", we perform path planning for exploration while also performing SLAM. Second, we propose a system specifically tailored to the inspection of wind turbines. Using an unmanned aerial vehicle equipped with 2D laser scanners, our method allows a quadcopter to take off, climb up the tower and inspect the front facing surface of the turbine's blades.

Both of our systems are implemented and shown to work within simulation environments. Our ground based inspection is also demonstrated in real indoor experiments where we show how to deal with noise not accounted for in simulation. Finally, we also show preliminary results on the deployment of our wind turbine inspection system.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

	English	Français
ASIC	Application Specific Integrated Circuit	Circuit Intégré à Application Dédiée
EP	Perimeter Exploration	Exploration du Périmètre
EC	Cavity Exploration	Exploration des Cavités
GIS	Geographical Information System	Système d'Information Géographique
GNSS	Global Navigation Satellite System	Système Mondial de Satellites de Navigation
GPS	Global Positioning System	Système de Positionnement Global
IMU	Inertial Measurement Unit	Centrale Inertielle
INS	Inertial Navigation System	Système de Navigation Inertiel
MVS	Multi-View Stereo Reconstruction	Reconstruction Stereo par Vues Multiples
NIR	Near Infrared	Quasi Infrarouge
PCA	Principal Component Analysis	Analyse des Composantes Principales
SFM	Structure From Motion	Structure par Mouvement
SLAM	Simultaneous Localization And Mapping	Locatisation et Cartographie Simultanée
UAV	Unmanned Aerial Vehicle	Véhicule Aérien Non-Habité
UGV	Unmanned Ground Vehicle	Véhicule Terrestre Non-Habité

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Le domaine de la robotique s'applique à ce qui est communément appelé les trois *D* de l'industrie *Dull*, *Dirty* et *Dangerous*, c'est-à-dire les tâches ennuyantes, sales et dangereuses. L'inspection d'infrastructures civiles rentre dans la première et la dernière de celles-ci. Ennuyante car elle implique la cueillette répétitive de données et dangereuse car elle implique parfois l'obligation de grimper dans des structures hors de portée ou difficiles à atteindre.

En général, le but de ces inspections est multiple : outre la recherche de défauts dans l'infrastructure, elles peuvent aussi servir à suivre de près l'évolution d'un chantier de construction, faire de l'arpentage ou planifier de futurs projets de développement. L'intérêt de faire exécuter l'inspection par un robot autonome est d'une part d'accélérer la collecte de données par un robot plus mobile qu'un humain, et d'autre part de réduire les coûts liés à la collecte d'informations.

1.1 Objectif de recherche

L'objectif principal de la recherche est de développer des méthodes d'inspection d'infrastructures adaptées aux robots terrestres non-habités (UGV) ainsi qu'aux robots aériens non-habités (UAV). La définition d'inspection peut varier d'un domaine à l'autre mais elle peut se résumer au déplacement d'un capteur quelconque pour recueillir de l'information sur l'entièreté d'un certain espace. Dans notre cas, nous traitons spécifiquement des scénarios où le capteur est une caméra ou une caméra de profondeur devant être déplacée pour capter la surface d'une structure et en reconstruire en modèle 3D.

Alors que la navigation du robot se fait entièrement de façon autonome, il est attendu qu'un opérateur intervienne seulement lors de la phase d'analyse des données et dans le cas d'un UAV qu'il intervienne dans la prise de photos.

1.1.1 Inspections au sol

En premier lieu, nous nous attardons au problème de la création de cartes 3D précises de structures au sol. Ces cartes peuvent être utilisées pour une variété d'applications telles que la réalité virtuelle (Google, 2017), la navivation de véhicules autonomes (DeepMap, 2017) et la surveillance du progrès d'un chantier de construction (Omar et Nehdi, 2018). Cette dernière application, étant bien établie en industrie, se fait habituellement par l'une des trois méthodes suivantes : la photogrammétrie ou vidéogrammétrie (Brilakis *et al.*, 2011), les scanner lasers

(Turkan *et al.*, 2012) ou même des caméras installées sur des véhicules aériens non-habités (Lin *et al.*, 2015). Chacune d'entres elles demande qu'un opérateur humain parcourt le chantier en cherchant les meilleurs points de vues avec les capteurs appropriés en main pour la collecte de données. En particulier, dû à la relative courte portée de capteurs d'images de profondeur, elles demandent aux opérateurs de parcourir une plus grande distance et plus de prises de vue.

C'est dans ce contexte que nous posons notre objectif de recherche qui est de guider un UGV pour faire l'inspection de la partie visible d'une structure au sol, au moyen d'un capteur de profondeur, sans aucune information *a priori*. Pour commencer, nous nous basons sur des algorithmes de navigation et de localisation existants tels que les champs de potentiel, le Simultaneous Localization and Mapping (SLAM) visuel et la fusion d'odométrie de roues avec une centrale inertielle (IMU) pour effectuer le contrôle du véhicule. L'objectif est d'avoir une méthode de génération de trajectoires en temps réel qui, dans un premier temps, garantit la couverture entière de la surface de l'édifice, et dans un deuxième temps assure la précision du système de SLAM. Cet objectif secondaire complémente l'objectif primaire car la précision de la localisation a un effect direct sur la qualité de reconstruction du modèle.

1.1.2 Inspections aériennes

Deuxièmement, nous considérons le cas où un UAV fait l'inspection de la surface d'une éolienne. Cette partie du projet, réalisée conjointement avec une compagnie privée opérant dans le secteur des véhicules aériens non habités, s'inscrit dans le cadre d'un plus large projet de recherche qui vise à développer une solution complète d'inspection d'éoliennes par UAV autonome permettant de réduire le temps nécessaire à une inspection. Tout d'abord, une inspection sert à repérer des défauts dans la structure d'une éolienne. Laissés seuls, ces défauts peuvent se traduirent en perte d'efficacité et dans le pire cas, en un bris fatal.

En temps normal, une inspection implique qu'un employé fasse tourner les pales de l'éolienne pour en positionner une vers le bas. L'employé grimpe en haut de la tour pour ensuite faire une descente en rappel le long de la pale pour inspecter visuellement et tactilement la surface de la pale. Ce processus devant être répété pour chaque pale, il peut prendre entre 2 et 4 heures dépendamment de la quantité de défauts à documenter. Plus récemment, nous notons un immense essor dans le nombre de compagnies offrant des services d'inspection par UAV qui ne requièrent pas l'ascention de la tour. Étant plus rapide, plus sécuritaire et moins dispendieuse, cette méthode est rapidement adoptée par l'industrie de l'énergie. Un rapport publié en 2015 par la firme d'études de marché Navigant Research prédit que le revenu global lié aux ventes d'UAV et de services d'inspection atteindra près de 6 milliards USD en 2024

(Navigant Research, 2015).

Le problème de ces systèmes est qu'ils demandent habituellement une équipe d'au moins deux personnes dont un pilote habile et un opérateur de caméra pour fonctionner. Le but général du projet est de rendre l'utilisation de l'UAV plus accessible afin de retirer le besoin d'un pilote et réduire la taille de l'équipe à une seule personne. Il faut noter que le but n'est pas encore de remplacer complètement une inspection manuelle par un humain. En effet, puisque la peinture reposant sur la surface d'une pale est moins flexible que le composite de la structure qui se retrouve en-dessous, un défaut dans la structure implique une fissure dans la peinture. L'inverse n'étant pas nécessairement vrai, une inspection visuelle par UAV demandera un deuxième passage de près par un humain pour confirmer l'existence du défaut et éliminer toute fausse alarme.

En somme, le but final du projet est de produire un système autonome plus facile d'utilisation pour l'inspection d'éoliennes et pouvant être déployé par un seul opérateur.

1.2 Plan du mémoire

Avant tout, nous commençons par une revue de littérature dans le Chapitre 2 pour passer en revue quelques concepts importants se rapportant à la vision par ordinateur et la navigation pour robots autonomes. Par la suite, nous présentons l'essentiel de notre travail en trois volets. Dans le Chapitre 2.7 nous faisons l'étude des capteurs existant sur le marché avec une comparaison des avantages et des désavantages de ceux-ci par rapport à l'inspection d'infrastructure. Au Chapitre 3, nous présentons notre travail réalisé dans le cadre de nos recherches sur l'inspection d'infrastructures par UGV qui s'est culminé en un article soumis en mars 2016 au journal *Transactions on Automation Science and Engineering*. En dernier lieu, au Chapitre 4, nous présentons notre travail réalisé lors du projet Mitacs Accelerate IT08812 en collaboration avec la compagnie Microdrones à propos de l'inspection d'éoliennes par drone autonome.

CHAPITRE 2 REVUE DE LITTÉRATURE

Bien que les deux types d'inspections traités dans ce mémoire sont exécutés sur des véhicules entièrement différents, les concepts utilisées lors de ces opérations se recoupent énormément. Dans ce chapitre, nous présentons d'abord dans la section 2.1 le travail antérieur directement en lien avec la génération de trajectoires d'inspection. Par la suite, nous traitons du sujet plus spécifique d'inspection d'éoliennes au moyen de robots aériens autonomes dans la section 2.2. Ensuite nous présentons les théories et les méthodes connexes requises pour l'opération des robots dans des environnements inconnus, en particulieur les méthodes de positionnement dans la section 2.3 et les méthodes de localisation et cartographie simultanée dans la section 2.4. Dans la section 2.5 nous survolons les méthodes communes de représentation d'environnement et dans la section 2.6 nous examinons la théorie importante pour le traitement d'images et la vision 3D. Finalement, nous terminons la revue de littérature par un survol des capteurs couramment utilisés pour la navigation de robots autonomes dans la section 2.7.

2.1 Génération de trajectoires d'inspection et couverture de surfaces

En général, le but des générateurs de trajectoires est d'optimiser une certaine métrique permettant de décider à quel endroit il faut placer le véhicule et son capteur pour faire l'inspection de la structure. Tout d'abord, considérons le cas d'une mission sans information *a priori*. Il s'agit d'une mission d'exploration où la génération de trajectoires doit se faire itérativement en temps réel pour envoyer le robot aux confins de l'espace connu. On tente en fait de répondre à la question :

> *Given what you know about the world, where should you move to gain as much new information as possible ?* (Sachant ce que nous savons à propos du monde, où devrions nous aller pour gagner le plus d'information possible ?) (Yamauchi, 1997).

Pour ce faire, Yamauchi introduit le concept de *frontier-based exploration* qui cherche à guider un robot vers la frontière entre l'espace connu et libre, et l'espace inconnu. Suivant cette idée plusieurs chercheurs proposent des fonctions de coût à minimiser permettant de choisir à quel endroit de la frontière explorer. Wirth et Pellenz (2007) proposent de subdiviser une carte 2D en cellules pour lesquelles une fonction de coût est calculée en prenant en compte non seulement la plus courte distance par rapport au robot mais aussi la distance par rapport à l'obstacle le plus proche. Chaque nouvelle position objectif minimise ainsi la distance pour

s'y rendre mais aussi le risque encouru par le robot. Au lieu d'utiliser les frontières comme objectifs Dornhege et Kleiner (2011) proposent une formulation du problème utilisant les frontières en tant que candidates possibles d'ouvertures dans les murs qui permettraient à une caméra de cartographier l'espace vide se trouvant derrière.

Bircher *et al.* (2016) proposent une méthode d'exploration 3D basée sur l'algorithme *Rapidly exploring Random Trees* (RRT) où un arbre est construit dans l'espace ouvert connu. À chaque sommet, l'angle de vue de la caméra est utilisé pour estimer le gain exploratoire de la position. Une fois l'arbre construit, le véhicule exécute la première étape de la branche possédant le plus grand gain total. L'arbre complet est ensuite recalculé en utilisant la branche choisie comme point de départ. En d'autres mots, leur méthode tente de maximiser le gain exploratoire en prenant aussi en compte les gains futurs d'une trajectoire. Outre les gains exploratoires, certains auteurs tentent aussi de prendre en compte l'effet des trajectoires sur leurs systèmes de navigation. Par exemple Papachristos *et al.* (2017) améliorent la méthode de Bircher en sélectionnant une trajectoire qui permet aussi de minimiser l'incertitude sur leur système de cartographie et d'odométrie visuelle. Wirth et Pellenz (2007) notent d'ailleurs que la proximité d'un obstacle est un danger (de collision) mais l'éloignement des obstacles en est aussi puisque la portée limitée des capteurs pourrait rendre un système de SLAM temporairement aveugle.

Alors que les méthodes précédentes se préoccupent de maximiser le gain d'information, elles ne prennent pas en compte les contraintes de temps liées à l'autonomie des robots, un problème qui affecte grandement les véhicules aériens multi-rotors. En combinant une fonction d'entropie calculée dans un voisinage local avec le coût en distance d'une trajectoire, Wang *et al.* (2017) proposent une méthode d'exploration se basant sur les *Information Potential Fields* (les champs de potentiels d'information). Similaire à la méthode des champs de potentiels pour l'évitement d'obstacles, le robot finit par être attiré dans les régions les plus proches à haut gain d'information. Toutefois, minimiser la longueur de la trajectoire n'assure pas nécessairement une minimisation du temps d'exploration si nous prenons aussi en compte l'accélération ou le jerk requis pour exécuter celle-ci. Pour résoudre ce problème Cieslewski *et al.* (2017) proposent une extension de la méthode de Yamauchi (1997) où la prochaine frontière choisie est sélectionnée dans le champ de vision du véhicule de telle sorte qu'il minimise le changement de vitesse requis pour le robot. Ainsi, la trajectoire exécutée peut être plus longue que les méthodes conventionnelles mais elle permet de maintenir une vitesse de navigation plus élevée.

Il est toutefois important de noter que ces méthodes d'inspection basées sur une variante de l'exploration de frontières ne se préocuppent pas d'assurer la couverture totale d'une

structure spécifique mais elles cherchent plutôt à explorer un certain espace ou un volume. Les méthodes d'inspection que nous proposons visent spécifiquement l'inspection de la surface atteignable d'une structure fermée.

Dans un autre ordre d'idée, une trajectoire complète et globalement optimale peut être générée au préalable si de l'information *a priori* est disponible. Pour une mission où la géométrie de la structure à inspecter est connue, Bircher *et al.* (2015) proposent de résoudre le problème en deux temps. En subdivisant le modèle en un maillage de triangles, on obtient pour chaque face un ensemble de points de vues admissibles. À chaque itération, des points de vue sont choisis séquentiellement en résolvant un problème d'optimisation QP (*Quadratic Programming*) où la fonction de coût minimise la distance par rapport aux points de vue voisins et où les contraintes assurent que la surface soit visible par le véhicule. Une fois l'ensemble choisi, une recherche heuristique est effectuée pour résoudre un problème de type commis voyageur à travers l'ensemble. Bircher réexécute les deux étapes jusqu'à ce qu'une trajectoire satisfaisante soit trouvée. Englot et Hover (2010) proposent un approche similaire de recherche par échantillonnage étant donné qu'une exploration exhaustive de l'espace des configurations possibles permettant la couverture du modèle serait excessivement long. Un graphe de poses possibles est construit et la trajectoire à générer est trouvée en résolvant un problème du facteur. À la différence d'un problème du commis voyageur où le but est de visiter chaque sommet, le but du problème du facteur est de visiter chaque arrête. La différence particulière de la méthode d'Englot est que l'observation du capteur est stockée dans les arrêtes du graphe et non le sommets.

Sheng *et al.* (2008) utilisent un modèle CAD d'un aéronef pour planifier une trajectoire permettant d'inspecter tous les rivets à la surface du véhicule. Yoder et Scherer (2016) débutent plutôt avec une boîte de délimitation 3D autour de la structure pour ensuite faire une exploration par frontière de surface.

Comparativement à l'inspection par exploration (en temps réel) la génération de trajectoires hors-ligne permet de créer des plans optimaux en termes de temps et de distance à parcourir. Par contre, cette trajectoire peut avoir un risque de collision s'il s'avère que le modèle utilisé diffère de la structure réelle, par exemple dans la cas d'un édifice partiellement endommagé par un désastre naturel ou une variation dans l'angle des pales d'une éolienne. Hepp *et al.* (2017) présentent un système faisant une première passe à haute altitude pour construire une carte rudimentaire dans laquelle la seconde inspection de près est planifiée suivant une maximisation du gain d'information.

Tel que mentionné brièvement précédemment, le travail que nous effectuons est aussi relié au problème de la planification de trajectoires pour la couverture de surfaces qui, par le passé, a

surtout été étudié dans le domaine 2D et supposent normalement un système de localisation précis (Hert *et al.*, 1996; Acar *et al.*, 2002; Acar et Choset, 2002; Lim *et al.*, 2014). Nous soulignons que notre contribution n'offre pas d'avancement théorique de ces méthodes de couverture. Nous proposons plutôt des solutions pratiques permettant de mettre en œuvre ces systèmes dans des situations réelles en présence de bruit de localisation et de capteurs.

2.1.1 *Active* SLAM et l'exploration sous incertitude

Le travail que nous présentons au niveau de l'inspection d'édifices par UGV est aussi étroitement lié aux domaines de l'*Active* SLAM et de l'exploration sous incertitude. Galceran *et al.* (2014) proposent une méthode similaire à celles présentées précédemmennt utilisant de l'information *a priori*. Par contre, ils ne supposent pas une exécution de trajectoire parfaite et considèrent l'incertitude sur la position du véhicule. Lors de l'inspection, une replanification de trajectoire basée sur l'optimisation stochastique est effectuée pour prendre en compte la vraie structure perçu en temps réel.

Kim et Eustice (2015) expliquent que l'*Active* SLAM est la planification de trajectoire en parallèle au SLAM. Dans le cas spécifique de Kim, ils introduisent le concept de *perception-driven navigation* (PDN) une façon d'équilibrer la les objectifs entre la minimisation de l'erreur sur le SLAM, le contrôle optimal et l'exploration de pour la couverture de surface. Dans le même ordre d'idées Stenning *et al.* (2013) proposent la construction en temps réel de réseaux de trajectoires réutilisables. Alors que le robot explore, les trajectoires empruntées sont gardées en mémoire pour construire un réseaux de chemins accessibles. La réutilisation de ces chemins permet ensuite de minimiser l'erreur de localisation lors de l'exploration. Stachniss *et al.* (2005) effectuent de l'*Active* SLAM au moyen de filtres particulaires alors que l'exploration se fait de façon similaire à Kim et Eustice (2015) en équilibrant le coût d'une action, l'incertitude de la carte, le gain exploratoire et les mesures des capteurs possibles le long de la trajectoire prévue.

2.2 Méthodes automatiques d'inspection d'éoliennes

L'inspection d'éoliennes par UAV autonome étant une application relativement nouvelle, peu de travaux ont été publiés sur ce sujet particulier et à ce jour, la majorité du travail accompli demeure théorique, en simulation seulement ou en environnement à petite échelle. Avant tout, Zhang et Jackman (2014) démontrent la faisabilité de repérer automatiquement des fissures sur la surface d'une éolienne au moyen de détecteurs de bordures bien connus comme les détecteurs Canny et Sobel. Un travail préliminaire a été réalisé sur les éléments de base requis

pour faire une inspection autonome. Notamment au niveau de l'utilisation du principe du flux optique pour l'estimation de la vitesse relative entre un véhicule aérien et une éolienne proposé par Høglund (2014) ainsi que l'utilisation de la transformée de Hough linéaire et de traitements géométriques pour la reconnaissance automatique d'éoliennes proposé par Stokkeland *et al.* (2015). Stokkeland explique qu'une inspection entièrement visuelle est difficile à réaliser dû à la grande quantité de bruit et de sources d'erreurs présentes sur le terrain. Par exemple, la segmentation entre la surface blanche de l'éolienne et les nuages en arrière-plan est extrêmement sensible au réglage des paramètres du filtre de couleur. C'est pourquoi Heggem (2017) fait usage de vision active au moyen d'un projecteur laser et de caméras stéréo pour détecter la distance et la forme de la pale. Avec une projection de seulement 11×11 points, Heggem parvient ensuite à calculer dans quelle direction son UAV doit se diriger pour poursuivre son inspection.

Du côté commercial, peu de compagnies se sont aventurées dans le domaine [1] mais toutes semblent faire usage de radars lasers et de caméras sous une forme ou une autre.

2.3 Méthodes de positionnement

Selon Borenstein *et al.* (1997), les systèmes de positionnement se regroupent dans l'une des deux catégories suivantes :

1. Les systèmes absolus fonctionnant grâce à de l'information externe tel que les GNSS reposant sur les données de satellites.
2. Les systèmes à l'estime fonctionnant par l'intégration d'une mesure à travers le temps tels que l'odométrie provenant des roues d'un robot ou la double intégrale d'une accélération pour estimer une position.

2.3.1 Systèmes de positionnement inertiel et par GNSS

Les systèmes de navigation complets vont habituellement se doter d'une combinaison des deux types de capteurs pour obtenir une estimation de la pose (position et orientation) du robot dans l'espace. Les systèmes de positionnement absolus tel que le GPS ayant normalement des fréquences de mise à jour lentes, autour de 1 Hz, sont souvent jumelés à une centrale inertielle (IMU), c'est-à-dire une combinaison de gyroscopes et d'accéléromètres permettant de mesurer la vitesse angulaire et l'accélération d'un corps rigide (Noureldin *et al.*, 2013). Bien que les capteurs inertiels permettent d'estimer la position à de hautes fréquences, parfois

1. Nous notons la présence de SkySpecs aux États-Unis, Pro-Drones en Italie et Perceptual Robotics en Angleterre.

même 1 kHz, en prenant l'intégrale de ceux-ci, une accumulation d'erreur peut rapidement faire diverger l'estimation d'état. Ainsi, un système inertiel à l'estime et un système GPS absolut jouent des rôles complémentaires où le système inertiel calcule la position à haute fréquence puis une correction est faite par GPS à basse fréquence.

On appelle tout système reposant sur des capteurs inertiels un Système de Navigation Inertiel (INS). De nos jours, ceux-ci sont faciles d'accès et peuvent êtres achetés pour quelques milliers de dollars. La fusion de ces différents capteurs peut se faire au travers de différents moyens tels que les filtres particulaires (Carvalho *et al.*, 1997) ou les graphes de facteurs (Indelman *et al.*, 2012) mais l'outil normalement utilisé est le filtre de Kalman (Noureldin *et al.*, 2013).

Le filtre de Kalman (KF) dans sa forme basique est un algorithme récursif permettant de faire une estimation aux moindres carrés d'un vecteur d'état au travers d'une procédure de prédiction de l'évolution du vecteur d'état suivant un modèle du système, suivi de la correction de celui-ci par les mesures entrant dans le filtre. Cet algorithme fonctionne pourvu que la transformation entre l'état et les mesures en entrée soit linéaire. Dans le cas d'un système non linéaire, une linéarisation par un développement de Taylor autour de l'estimé optimal courant du vecteur d'état est effectuée, résultant en un filtre de Kalman étendu (EKF) (Chui et Chen, 2017).

Les GNSS n'étant pas la seule façon de corriger l'erreur cumulative d'un INS, certains auteurs proposent des filtres de Kalman modulaires permettant de fusionner une quantité arbitraire de capteurs hétérogènes tels que des IMU, l'odométrie d'encodeurs de roues, l'odométrie visuelle, receveurs GNSS, etc. (Moore et Stouch, 2014) pour estimer la position et l'orientation du robot. Pour le cas spécifique des UAV, Lynen *et al.* (2013) proposent aussi une architecture modulaire pour un filtre de Kalman étendu itéré qui permet de faire une calibration inter-capteurs en ligne pour estimer les délais temporels et les différences d'échelle des mesures, chose importante dans le cas d'odométrie visuelle monoculaire où l'échelle des mesures est arbitraire et peut dériver avec le temps.

2.4 *Simultaneous Localization and Mapping*

Les GNSS permettent de localiser un robot sur la surface de la Terre, mais qu'en est-il des situations dépourvues de réception satellite adéquate telles que les environnements intérieurs ou les « canyons urbains » ? De plus, nous sommes souvent plus intéressés à connaître la position du robot par rapport à son environnement plutôt que sa latitude et sa longitude. Nous cherchons donc à répondre à « Comment localiser un robot dans son environnement sans une carte de celle-ci ? » ; Une question qui est l'essence même du problème de localisation

et cartographie simultanée (SLAM).

Un système de SLAM cherche donc à créer une représentation de l'environnement pour ensuite s'y localiser. Cette représentation peut prendre plusieurs formes dépendamment du capteur utilisé mais elle cherche habituellement à extraire des points de repère de l'environnement pour former une carte dans laquelle le robot sera localisé. Pour une caméra RGB, ces points de repère pourraient être des caractéristiques visuelles extraites des images (Mur-Artal et Tardós, 2017a). Une particularité des systèmes de SLAM entièrement visuel et monoculaire est que la carte construite sera à une échelle arbitraire et ne peut donc pas être utilisée pour la navigation d'un robot. Ces dernières années beaucoup de travail a été réalisé pour combiner une caméra monoculaire à une centrale inertielle pour résoudre cette ambiguïté d'échelle (Mur-Artal et Tardós, 2017b). Une autre façon d'éviter le problème est d'utiliser une caméra RGB-D ou une caméra stéréo où la forme des objets en plus des caractéristiques visuelles peut être utilisée pour cartographier (Henry *et al.*, 2014). Zhang et Singh (2017) montrent que du SLAM peut aussi être réalisé grâce à un lidar 3D en faisant l'extraction de caractéristiques planaires et de contours.

Bref, une multitude d'approches existent pour l'extraction des points de repère mais ces données seules ne sont pas suffisantes pour la localisation du robot. Les données captées doivent ensuite être traitées par un quelconque mécanisme d'estimation d'état. Grisetti *et al.* (2010) expliquent que les systèmes se séparent habituellement en deux catégories, ceux par filtrage et ceux par optimisation (Grisetti réfère au *lissage*). Les systèmes par filtre cherchent à estimer l'état courant du robot par rapport à la carte et peuvent utiliser par exemple un filtre particulaire (Grisetti *et al.*, 2007) ou un EKF (Montemerlo *et al.*, 2003). En revanche, les méthodes par optimisation cherchent à estimer la trajectoire complète du robot au moyen de l'ensemble complet des observations qu'il a recueillies. Pour ce faire, il est intuitif de modéliser le problème de SLAM par un graphe de poses où les connexions entre les noeuds représentent des observations de l'environnement. Ces observations imposent donc des contraintes sur le graphe sur lequel nous cherchons à effectuer une minimisation de l'erreur suivant une formulation par moindres carrés. Cette méthode est utilisé par plusieurs systèmes de SLAM proposés récemment (Labbé et Michaud, 2014; Hess *et al.*, 2016).

Une composante importante de tout système de SLAM est sa capacité de fermeture de boucle, c'est-à-dire d'être capable de reconnaître quand le robot retourne à un endroit déjà visité, particulièrement en l'absence de systèmes de positionnement absolus. Ceci peut être fait localement sur un sous-ensemble local des observations courantes ou globalement sur toutes les observations faites jusqu'au présent. Ceci peut être réalisé d'une variété de façons dépendamment des capteurs utilisés par l'algorithme de SLAM. Hess *et al.* (2016) proposent

Google Cartographer une approche hybride fonctionnant par scanner laser, où les nouveaux balayages sont insérés et appariés par rapport à une sous-carte locale alors que la recherche de fermeture de boucle globale est exécutée en arrière-plan par un algorithme de séparation et d'évaluation progressive (*branch-and-bound*). En revanche RTAB-Map (Labbé et Michaud, 2014) utilise une approche par sac-de-mots (*bag-of-words*) appliquée à des descripteurs extraits d'une image couleur. Pour assurer une opération en temps réel, les nœuds du graphe de poses, contenant aussi les mots visuels extraits à chaque endroit, sont séparés dans différentes mémoires : la mémoire court terme, la mémoire de travail et la mémoire long terme. Seuls les noeuds dans la mémoire de travail sont considérés pour la fermeture de boucle. Dans les deux cas de Google Cartographer et RTAB-Map, il est donc possible que la détection de la fermeture de boucle échoue si la pose estimée courante du robot a trop dérivé au point que les algorithmes ne cherchent plus à apparier l'environnement courant avec les éléments de la carte connue.

Le but de la fermeture de boucle est de réduire les erreurs de localisation en imposant certaines contraintes sur la carte en cours de construction. Cette minimisation d'erreur peut être réalisée de plusieurs façons, notamment par optimisation de graphe de poses (Carlone *et al.*, 2016), par compensation par faisceaux (*Bundle Adjustment*) (Mei *et al.*, 2011) ou tel que proposé dans ORB-SLAM2 une combinaison des deux (Mur-Artal et Tardós, 2017a).

2.5 Méthodes de représentation de l'environnement

Pour faire naviguer un robot mobile dans un environnement inconnu, il est impératif d'avoir une représentation de l'environnement dans laquelle la planification de trajectoire et la vérification de collision est possible. L'important est d'utiliser la représentation appropriée correspondant aux mouvements possibles du robot. Pour la navigation en 3D, le cadriciel OctoMap de Hornung *et al.* (2013) est particulièrement populaire par son code source ouvert, sa représentation probabiliste de l'environnement connu et inconnu, et son efficacité en termes de mémoire. Une OctoMap représente l'espace volumique par un Octree, subdivisant ainsi récursivement l'espace par un arbre d'octants. Chaque niveau de l'arbre représente une résolution différente de l'espace tel qu'on peut le voir dans la Figure 2.1.

La structure Octree rend l'OctoMap particulièrement efficace en termes de mémoire comparativement à une séparation naïve de l'espace en voxels car l'arbre n'est étendu qu'au besoin. Alors que certaines représentations de l'espace utilisent une forme binaire où tout voxel est soit occupé ou libre, OctoMap utilise une forme probabiliste où chaque voxel est inconnu ou a une probabilité d'être occupé ou libre permettant donc à l'OctoMap de représenter l'incertitude et le bruit de mesure sur le capteur utilisé pour construire la carte. La probabilité

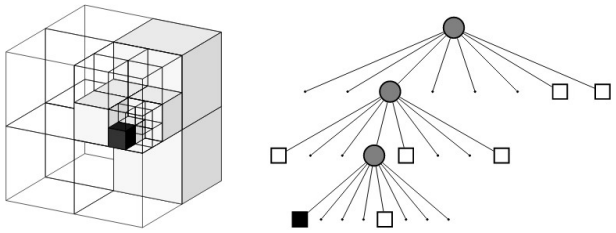


Figure 2.1 L'octree est une structure en arbre souvent utilisé pour partitionner un espace tridimensionnel. Figure prise de Hornung *et al.* (2013).

$P(n|z_{1:t})$ qu'une feuille n de l'Octree soit occupée étant donné les mesures de capteurs $z_{1:t}$ est calculé selon

$$P(n|z_{1:t}) = \left[1 + \frac{1 - P(n|z_t)}{P(n|z_t)} \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1} \tag{2.1}$$

avec la probabilité par défaut $P(n) = 0.5$. Les probabilités des voxels peuvent ensuite être dicrétisées pour définir l'espace vide ($P(n) < 0.12$), l'espace inconnu ($P(n) \in [0.12, 0.97]$) et l'espace occupé ($P(n) > 0.97$) pour appliquer divers algorithmes de génération de trajectoires tels que les RRT ou A*.

Par contre, une OctoMap n'est pas particulièrement bien adaptée aux méthodes de planification de trajectoires par optimisation qui ont besoin de connaître la distance aux obstacles en tout point de la carte ainsi que les gradients de distance (Ratliff *et al.*, 2009; Oleynikova *et al.*, 2016). Ce calcul pouvant être coûteux, Oleynikova *et al.* (2017) proposent d'utiliser une représentation des surfaces par des fonctions de distance signées tronquées offrant une accélération par la suite dans le calcul des champs de distance euclidiennes signées.

Dans un autre ordre d'idées, Fridovich-Keil *et al.* (2017) abandonnent complètement la représentation par grille et proposent d'utiliser des sphères (des «atomes ») organisés dans un arbre k-d. Ceci permet une représentation plus précise de l'espace libre tangent aux surfaces en plus de faire l'estimation de fonctions de distance signées en temps réel.

2.6 Traitement d'images

2.6.1 Modèle de caméra

Il arrive souvent dans des applications de traitement d'images de devoir passer du domaine 3D à 2D et vice-versa. Pour ce faire, il est important de passer en revue les modèles de projection et de distorsion d'images couramment utilisés. Tout point 3D $\mathbf{p}_w$ à coordonnées

connues peut être projeté sur le plan image en un point dont les coordonnées en pixels $\boldsymbol{x_s}$ sont données par l'équation 2.2.

$$\boldsymbol{x}_s = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{p}_w = \mathbf{P}\mathbf{p}_w \tag{2.2}$$

où $\mathbf{P} \in \mathbb{R}^{3\times4}$ désigne la matrice de caméra, $\mathbf{K} \in \mathbb{R}^{3\times3}$ la matrice des paramètres intrinsèques, $\mathbf{R} \in \mathbb{R}^{3\times3}$ est une rotation orthogonale et $\mathbf{t}$ une translation. Dépendamment des auteurs, $\mathbf{K}$ peut être posée différemment ; l'équation 2.3 présente sa forme générale où f_x et f_y désignent la distance focale en pixels, (c_x, c_y) le centre optique en pixels, s le *skew* prenant en compte l'angle possible entre les axes du capteur et les axes optiques, et finalement α le facteur de forme :

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & \alpha f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

En pratique la majorité des applications telles que la librairie de traitement d'images OpenCV vont simplifier $\mathbf{K}$ au moyen de $s = 0$ et $\alpha = 1$ (Itseez, 2017). Il est aussi utile de passer aux coordonnées homogènes au moyen d'une matrice $\tilde{\boldsymbol{P}} \in \mathbb{R}^{4\times4}$ en ajoutant une ligne à $\mathbf{P}$,

$$\tilde{\boldsymbol{P}} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \tilde{\boldsymbol{K}}\boldsymbol{E} \tag{2.4}$$

où $\boldsymbol{E}$ est une matrice de transformation 3D. La projection se fait de la même façon que l'équation 2.2 avec $\bar{\boldsymbol{p}}_{\boldsymbol{w}} = \begin{bmatrix} \mathbf{p}_w^\top & 0 \end{bmatrix}^\top$ et $\boldsymbol{x_s} = \begin{bmatrix} x_s & y_s & 1 & d \end{bmatrix}^\top$

$$\boldsymbol{x_s} \sim \tilde{\boldsymbol{P}}\bar{\boldsymbol{p}}_{\boldsymbol{w}} \tag{2.5}$$

où le symbole $\sim$ indique une égalité à l'échelle et $\boldsymbol{x_s}$ doit être normalisé après la multiplication pour que la troisième composante soit 1 (Szeliski, 2011).

Ces formules sont valides dans le cas de projections entièrement linéaires. En revanche, les systèmes optiques vont habituellement introduire une certaine distorsion de l'image, surtout dans le cas de lentilles grand angle.

Plusieurs modèles de distorsion ont été proposés à travers le temps mais le plus répandu est le modèle radial-tangentiel, radial puisqu'il dépend de la distance du pixel par rapport au centre optique, et tangentiel pour le déplacement des rayons tangents au cercle autour du centre optique. La distorsion radiale-tangentielle peut être exprimée par un polynôme permettant de mettre en correspondance les pixels de l'image originale (x_c, y_c) en coordonnées normalisées

aux pixels de l'image rectifiée (u, v) :

$$\begin{aligned}
\hat{x}_c &= x_c \left(\frac{1 + \kappa_1 r_c^2 + \kappa_2 r_c^4 + \kappa_3 r_c^6}{1 + \kappa_4 r_c^2 + \kappa_5 r_c^4 + \kappa_6 r_c^6} \right) + 2p_1 x_c y_c + p_2 (r_c^2 + 2x_c^2) \\
\hat{y}_c &= y_c \left(\frac{1 + \kappa_1 r_c^2 + \kappa_2 r_c^4 + \kappa_3 r_c^6}{1 + \kappa_4 r_c^2 + \kappa_5 r_c^4 + \kappa_6 r_c^6} \right) + p_1 (r_c^2 + 2y_c^2) + 2p_2 x_c y_c \\
r_c^2 &= x_c^2 + y_c^2
\end{aligned} \tag{2.6}$$

et

$$\begin{aligned}
u &= f_x \hat{x}_c + c_x \\
v &= f_y \hat{y}_c + c_y
\end{aligned} \tag{2.7}$$

Les paramètres $\kappa_{[1-6]}$ sont les coefficients de distorsion radiaux et $p_{[1,2]}$ les coefficients de distorsion tangentiels. L'équation 2.7 permet de mettre à l'échelle les coordonnées normalisées en coordonnées en pixels. Les paramètres de distorsion et intrinsèques peuvent être estimés conjointement suivant la résolution d'un problème des moindres carrés non-linéaire (Zhang, 2000).

Bien que le modèle radial-tangentiel soit le plus populaire, entres autres par sa présence dans la librairie de traitement d'images OpenCV, plusieurs auteurs proposent de nouveaux modèles avec divers avantages. Par exemple le modèle de distorsion équidistant proposé par Kannala et Brandt (2006) est apte à modéliser la distorsion à la fois de lentilles normales et de lentilles ultra-grand angle et le modèle par fonction rationnelle de Claus et Fitzgibbon (2005) adapté aux lentilles grand-angle et catadioptriques.

2.6.2 Vision stéréo

Figure 2.2 Exemple de caméra stereo PointGrey Bumblebee2

La vision stéréo est un sujet étudié depuis longtemps pour diverses utilisations tel que la cueillette d'objets par bras manipulateur (Hernandez *et al.*, 2017), l'effacement d'arrière-

plan (Kanade *et al.*, 1996) et la navigation de robots (Fraundorfer *et al.*, 2012). Sans aller dans les détails de la géométrie épipolaire permettant de trianguler un point dans l'espace, l'idée générale derrière la vision stéréo est de comparer les images de deux caméras pour calculer la distance entre deux points correspondants. Cette distance, que l'on nomme la disparité, est directement liée à la position 3D du point dans le monde selon la relation

$$d = f\frac{B}{Z} \tag{2.8}$$

où d est la disparité, f est la longueur focale en pixels, B est la distance entre les deux caméras et Z est la profondeur du point. Diverses méthodes existent pour mettre en correspondance les points (ou plus souvent des blocs de plusieurs pixels) entre les deux images. Nous avons par exemple les algorithmes locaux cherchant le plus petit coût de correspondance suivant une métrique telle que la différence absolue ou la somme des différences au carré calculée sur l'intensité des pixels d'un bloc (Szeliski, 2011). Ces coûts locaux peuvent aussi être optimisés globalement par *belief propagation* pour obtenir une image de disparité cohérente (Klaus *et al.*, 2006). L'approche utilisée dans la librairie de traitement d'images OpenCV est en fait une approximation de l'optimisation globale où la programmation dynamique est utilisé pour optimiser une fonction d'énergie (Hirschmuller, 2008).

Cependant, peu importe l'algorithme utilisé, il sera toujours important d'avoir de la texture dans l'image, sans quoi il existera une incertitude sur la correspondance des blocs et le calcul de disparité échouera. Outre les filtres lisseurs tentant de remplir les trous dans une image de disparité où la recherche de correspondance aurait échoué, les avancées récentes dans le domaine tendent vers l'utilisation de l'apprentissage machine pour prédire l'image de disparité en entier (Kendall *et al.*, 2017). Meier *et al.* (2017) proposent une solution plus classique où l'échec de correspondance est prédit et corrigé au moyen d'une caméra stéréo secondaire placée orthogonalement à la caméra principale.

2.6.3 Détection de lignes

La méthode la plus connue pour la détection de contours dans une image reste à ce jour celle proposée par Canny (1986) qui consiste brièvement en un calcul de gradient suivi d'une suppression des non-maxima. Avec la popularité grandissante de l'apprentissage machine, certaines nouvelles méthodes ont été proposées pour la détection de contours à base de réseaux de neurones. Par exemple, *Holistically-Nested Edge Detection* proposé par Xie et Tu (2015) est un réseau de neurones entièrement convolutionnel qui fait aussi usage d'apprentissage multi-échelle de caractéristiques imbriquées. L'avantage majeur des méthodes par apprentissage

machine est qu'elles ne requièrent pas la calibration de paramètres.

Une fois les contours détectés et inscrits dans une image binarisée, il devient possible d'appliquer une transformée de Hough pour la détection de lignes droites (Duda et Hart, 1972). Avec le temps, plusieurs améliorations ont été proposées pour accélérer le calcul de la transformée, notamment l'introduction de méthodes probabilistes n'utilisant qu'un sous-ensemble des données pour la détection des lignes (Matas *et al.*, 2000). Herout *et al.* (2013) dénombraient 39 variantes de la transformée de Hough incluant des variantes faisant usage de matériel spécialisé tel que des unités de traitement graphiques (GPU) ou des processeurs configurables (FPGA).

D'autres méthodes de détection de ligne ont aussi été proposées sans le besoin de passer par une transformée de Canny. Par exemple, le *Line Segment Detector* de Grompone von Gioi *et al.* (2012) fonctionne par régions de support de ligne. D'un autre côté, EDLines proposé par Akinlar et Topal (2011) fonctionne directement sur une image à tons de gris : des points d'ancrage sont calculés sur une image gradient pour effectuer un tracé de contours. LSD et EDLines ont tous les deux l'avantage d'être rapides et ont été conçus pour ne pas avoir besoin de réglage de paramètres.

2.7 Choix de capteurs

Il va de soi que pour qu'un robot puisse éviter des obstacles et se déplacer de façon autonome, il doit avoir un moyen de percevoir son environnement. Dans ce chapitre, nous traitons des capteurs utilisables dans le contexte de la navigation de robots mobiles, et possiblement de la reconstruction 3D. En général, il existe trois capteurs couramment utilisés : la vision passive, la vision active et les télémètres laser.

2.7.1 Systèmes de vision passive

Dans la section 2.6.2, nous avons touché au sujet de vision stéréo, qui est un exemple de premier plan d'un capteur passif, c'est-à-dire un capteur capable de percevoir un phénomène quelconque sans devoir émettre un signal. En effet, les caméras sont un moyen peu coûteux de percevoir la profondeur et sont couramment utilisées dans des micro-véhicules aériens commerciaux tel que le Yuneec Typhoon H de la Figure 2.3a. Par contre, la vision stéréo n'étant pas infaillible, la perception de profondeur peut échouer dans les environnements comportant peu de textures ou trop peu de lumière. Malgré certaines méthodes de régularisation pour tenter de remplir les trous en propageant l'information des régions aux alentours (Hirschmuller, 2008), l'état de l'art est à un point où les entreprises jugent encore prudent d'ajouter

un capteur secondaire, tel qu'un sonar dans la Figure 2.3b, pour compenser les échecs de perception.

(a) (b)

Figure 2.3 (a) Le multi-rotor Yuneec Typhoon H (b) La caméra stéréo infrarouge Intel RealSense augmentée d'un sonar pour détecter les obstacles lors des échecs de perception.

Dans le contexte des micro-véhicules aériens, il arrive parfois que l'on veuille miniaturiser et simplifier le système à un point où il serait désirable d'avoir un système monoculaire au lieu d'un système stéréo. Tel que mentionné dans la section 2.4, il est possible de mettre à l'échelle une carte 3D construite monoculairement au moyen d'une source d'information métrique secondaire tel qu'une centrale inertielle (Mur-Artal et Tardós, 2017b). Par contre, pour que cette carte soit utilisable pour la navigation, elle doit être construite de façon à pouvoir représenter les surfaces des obstacles. Yang *et al.* (2017) proposent de profiter de l'exactitude d'un système d'odométrie visuo-inertiel pour construire des cartes de profondeur au moyen de résolution stéréo par mouvement. En d'autres mots, au lieu d'avoir deux caméras pour la triangulation de points, nous avons une seule caméra qui se déplace dans l'espace pour la triangulation.

2.7.2 Systèmes de vision active

Une autre façon de compenser pour les lacunes de la vision stéréo est d'utiliser la vision active, c'est-à-dire de projeter une onde quelconque pour en détecter le retour. L'un des exemples les mieux connus de ce type de système est la ligne de produits Microsoft Kinect. Originalement prévue pour permettre aux usager des consoles de jeux Xbox360 de jouer à des jeux contrôlés par gestes, la caméra de profondeur a rapidement été adoptée par le milieu académique pour son faible coût et sa précision relativement élevée (Khoshelham et Elberink, 2012). La Kinect pour Xbox360 fonctionnait originalement au moyen d'un projecteur laser quasi-infrarouge (NIR) et d'une caméra NIR. Suivant le principe de la « lumière structurée »un motif NIR

connu est projeté sur la scène et une caméra analysant la déformation du motif permet de calculer la profondeur de la scène (Zhang, 2012).

(a) (b)

Figure 2.4 (a) Caméra Ensenso 3D fonctionnant par projection de texture et résolution stéréo. (b) Résultat de la méthode proposée par (Konolige, 2010).

Konolige (2010) propose une méthode similaire où une caméra stéréo est augmentée d'un projecteur de texture pour permettre la résolution de la profondeur dans les scènes dépourvues de texture adéquate. La mise en correspondance des blocs stéréo se fait avec la connaissance *a priori* de la texture qui avait été projetée. À ce jour, ce principe est utilisé dans une variété de produits commerciaux tels que la caméra Ensenso 3D de IDS Imaging.

Par contre, les caméras par lumière structurée souffrent de plusieurs inconvénients, notamment leur sensibilité à la lumière infrarouge ambiante et leur imprécision dans le cas de scènes en mouvement (Khoshelham et Elberink, 2012). Durant les cinq dernières années, nous avons assisté à une baisse de prix considérable dans les capteurs de profondeur fonctionnant par *time-of-flight (ToF)* (temps de vol). Le principe du ToF se résume à projeter périodiquement une lumière NIR modulée en intensité et d'en mesurer le temps de retour. Bien sûr, cette dernière n'est pas mesurée directement ; elle est plutôt calculée à partir du déphasage du signal de retour capté par la caméra. L'avantage de caméras ToF est qu'elles offrent une bonne performance dans des environnements à haute et basse luminosités, en plus d'offrir un temps de réponse rapide. Encore ici, l'exemple le plus répandu d'une caméra ToF est la Microsoft Kinect v2 pour Xbox One.

Une dernière amélioration récente dans le domaine des caméras de profondeur est la ligne de produits Intel RealSense (Figure 2.3b) fonctionnant par lumière non-structurée (*unstructured light*) dont le traitement est accéléré par un circuit intégré à application dédiée (ASIC). Une texture fixe est projetée sur la scène au moyen d'un projecteur laser infrarouge. Cette texture est précalculée pour maximiser le contraste et pour minimiser la similitude le long de l'axe

de recherche épipolaire. À l'inverse de la Kinect 1, la texture est inconnue au moment de la résolution stéréo et ne sert que d'aide secondaire au système. L'originalité de la solution provient du fait que la caméra stéréo infrarouge permet d'opérer une RealSense dans un environnement où la lumière du soleil est de forte intensité ainsi que dans la noirceur totale grâce au projecteur. De plus, la nature matérielle des caméras stéréo fait en sorte que la RealSense peut être produite à faible coût, dans un petit emballage et avec une consommation électrique relativement faible (Keselman *et al.*, 2017).

2.7.3 Télémètres laser

Un dernier type de capteur à considérer est le télémètre laser ou LIDAR pour *Light Detection and Ranging*. Opérant souvent par ToF, les télémètres laser sont combinés à des assemblages de moteurs et de miroirs pour obtenir des scanners laser en 2D ou 3D. L'un des exemples les plus médiatisés de l'utilisation de lidars pour la navigation de véhicules terrestres était Stanley, la voiture autonome de Stanford qui a gagné le DARPA Grand Challenge en 2005 (Thrun *et al.*, 2006).

Figure 2.5 Stanley était équipé de 5 lidar 2D sur son toit pour la détection d'obstacles et la navigation (Thrun *et al.*, 2006).

Avec le temps, les lidars se sont améliorés pour devenir plus compacts et moins dispendieux à un point où il est possible de les installer sur des véhicules multi-rotors pour fins de cartographie et de localisation (Zhang et Singh, 2018). La tendance de miniaturisation se maintient avec l'arrivée récente sur le marché de lidars à état solide sans moteurs ou miroirs pour la déflection du faisceau laser. Dans le chapitre 4, nous démontrerons l'utilisation d'un tel capteur, le LeddarVu8 pour la navigation d'un véhicule aérien.

(a) (b)

Figure 2.6 (a) Le Quanergy S3 avec 8000 points dans un angle de vue de 120° horizontal et vertical (Eldada, 2016). (b) Le LeddarVu8 avec 8 segments de détection.

CHAPITRE 3 INSPECTION D'INFRASTRUCTURES PAR UGV

Dans cette section, nous présentons notre travail réalisé dans le cadre de l'article *Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures* paru dans IEEE Transations on Automation Science and Engineering (Ramanagopal *et al.*, 2017). Le contenu diffère de l'article pour ajouter des explications à propos de certains sujets qui avaient été écourtés par manque d'espace et pour mettre plus d'emphase sur la partie d'implémentation de la méthode. Certaines figures produites en simulation ont aussi été remplacées par leurs équivalents en photo réalisées lors des expériences et plusieurs figures ont été ajoutées pour mieux illustrer certains phénomènes expliqués. La théorie a initialement été développée par M. S. Ramanagopal et notre contribution se situe au niveau de l'expérimentation et l'implémentation sur la vraie plateforme robotique. Cette expérimentation a permis de dénicher certaines lacunes au niveau des algorithmes et de la gestion du bruit de capteur que nous avons par la suite corrigées.

La section 3.1 met en contexte le problème que nous abordons, les sections 3.2 et 3.3 décrivent les phases de l'algorithme d'inspection et finalement la section 3.4 présente nos résultats de simulation et d'expérimentation.

3.1 Description du problème

Le but du projet présenté dans ce chapitre est de développer une stratégie permettant de guider un robot terrestre équipé d'une caméra ou d'un capteur de profondeur pour cartographier la partie visible d'une structure 3D. Nous décrivons une méthode de planification de trajectoire qui permet de générer des points de vues successifs pour parcourir le périmètre de la structure suivi d'une deuxième inspection dans laquelle nous cherchons à compléter le modèle en inspectant les trous provenant de cavités dans la structure.

Plus spécifiquement, notre contribution consiste en une méthode de génération de trajectoires qui, d'une part, assure une couverture complète de la surface de la structure, et qui d'autre part, facilite la précision de la reconstruction. Cette dernière est réalisée en tentant de faire une fermeture de boucle le plus tôt possible dans la trajectoire (voir sec. 2.4). La minimisation d'erreur de localisation par la fermeture de boucle se traduit directement en une amélioration de la précision de reconstruction du modèle 3D de la structure. Nous évaluons donc la performance de nos algorithmes selon la précision du modèle reconstruit et nous illustrons les avantages de notre méthode en la comparant à l'approche classique de l'exploration de

frontières. Dans le cas des expériences pratiques, nous n'évaluons que la trajectoire effectuée car nous n'avions pas de moyen de comparer le modèle reconstruit à la vérité-terrain. Nous notons par contre qu'il est possible d'obtenir la vérité terrain de la structure sous inspection au moyen de radars lasers industriels tel qu'un Leica ScanStation. De plus, la longueur de la trajectoire n'est pas une métrique que nous prenons en compte.

3.1.1 Hypothèses et fonctionnement de l'inspection

Hypothèse 1 L'inspection débute sans information au préalable à propos de la structure. La caméra à bord de l'UGV est placée à une hauteur fixe $(0, 0, h_c)$ par rapport à R et nous permet de recevoir une série de nuages de points combinée à des images RGB pour le système de SLAM visuel. Le système de SLAM nous fournit à son tour une carte 3D de l'environnement ainsi que notre trajectoire à travers celle-ci. Pour ce projet, nous faisons spécifiquement usage de RTAB-Map (Labbé et Michaud, 2014), car c'est un système de SLAM à source ouverte, mais nous soulignons que tout autre système de SLAM pourrait être utilisé, pourvu qu'il fournisse l'information énoncée précédemment et que la détection de fermeture de boucle soit possible. Au départ de l'algorithme, l'UGV pointe la caméra vers sa droite face à l'un des murs de la structure.

Hypothèse 2 Soit une distance de sécurité D devant être maintenue par rapport à la structure. Nous supposons que l'espace se situant à $2D$ de la structure est libre d'obstacles. Ceci nous permettra à la section 3.2 de savoir que les surfaces détectées par le LIDAR sont des extensions de la structure sous inspection.

Hypothèse 3 Finalement, nous supposons que la structure repose sur le plan horizontal $z^g = 0$. Ceci nous permet de simplifier certains de nos algorithmes pour filtrer le sol des nuages de points. Par extension, nous supposons aussi que la mission de l'UGV n'est que d'inspecter la partie de la structure visible dans le champ de vision de la caméra au sol. Ainsi, la hauteur maximale de la structure pouvant être cartographiée est $H_{max} = h_c + D \tan \psi/2$ où ψ est l'angle de vue vertical de la caméra.

L'inspection se fait en 2 phases : l'exploration du périmètre (EP) présentée dans la section 3.2 suivie de l'exploration des cavités (EC) présentée dans la section 3.3. Dans la première phase, l'UGV fait le tour de la structure en sautant les cavités pour fermer la boucle le plus tôt possible. Ensuite, une analyse de la carte 3D est effectuée pour trouver les frontières indiquant des cavités non explorées dans la structure. Une trajectoire est planifiée pour diriger le robot vers la cavité pour cartographier l'intérieur de la structure. L'inspection se termine quand toutes les cavités ont été inspectées.

Tableau 3.1 Repères présents dans le système d'inspections par UGV

Symbole	Nom	Description
$\mathsf{G} := \{\mathsf{O}_\mathsf{G}, \mathbf{x_g}, \mathbf{y_g}, \mathbf{z_g}\}$	*Global*	L'origine O_G se situe au point de départ du robot et les axes $\{\mathbf{x_g}, \mathbf{y_g}, \mathbf{z_g}\}$ sont alignés selon la pose initiale du robot.
$\mathsf{R} := \{\mathsf{O}_\mathsf{R}, \mathbf{x_r}, \mathbf{y_r}, \mathbf{z_r}\}$	*Robot*	Repère centré sur le robot avec les axes avant-gauche-haut.
$\mathsf{C} := \{\mathsf{O}_\mathsf{C}, \mathbf{x_c}, \mathbf{y_c}, \mathbf{z_c}\}$	*Camera*	Repère centré sur la caméra, la transformée est rigide par rapport à R sauf pour l'angle de lacet.

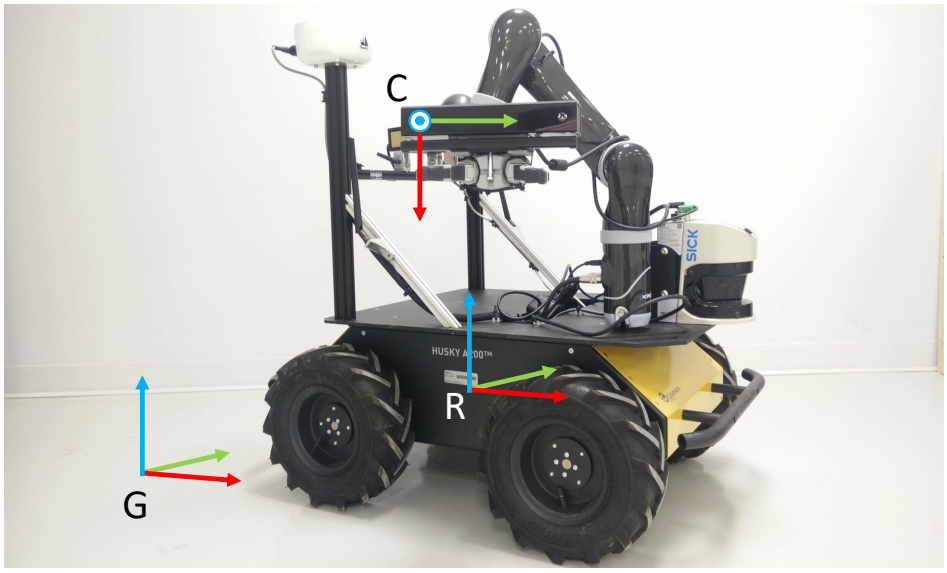


Figure 3.1 Visualisation des repères présents sur le système d'inspection par UGV.

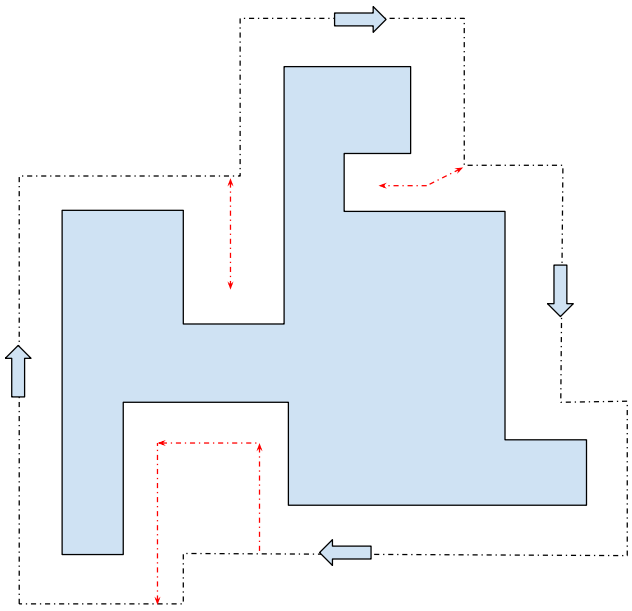


Figure 3.2 Illustration du but de l'algorithme. La périmètre trajectoire en noir indique l'exploration du périmètre qui saute les cavités pour revenir au point de départ le plus tôt possible et fermer la boucle. Les trajectoires rouges de la phase d'exploration de cavités viennent par après une fois que suffisamment de contraintes ont été mises sur la carte.

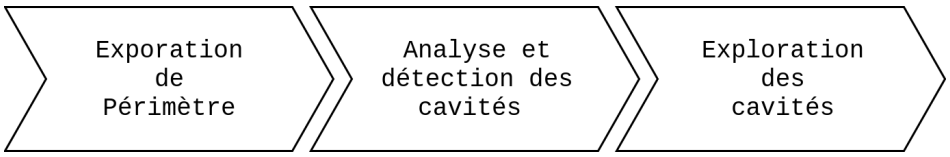


Figure 3.3 Diagramme de séquence des phases de l'inspection.

3.2 Exploration du périmètre

Suivant les hypothèses de la section 3.1.1, le problème d'inspection se simplifie au parcours de la partie de la structure située entre les plans $z^g = 0$ et $z^g = H_{max}$. Pour ce faire la phase EP débute avec le robot près de l'un des murs de la structure. Le robot parcourt le périmètre dans le sens horaire gardant toujours une partie visible de la structure dans son champ de vision vers sa droite. À chaque itération de l'algorithme, l'UGV détermine la prochaine pose objectif à laquelle se rendre pour placer la caméra à une distance D perpendiculairement à la surface de la structure, maximisant ainsi la qualité des détails perçus et la densité des points captés.

3.2.1 Choix de la prochaine pose objectif

L'algorithme 1 présente la méthode par laquelle nous calculons la prochaine pose objectif, permettant à l'UGV de suivre la surface de la structure. Une visualisation du résultat de l'algorithme est disponible dans l'annexe A.

Algorithme 1 : Calcul de la prochaine pose objectif au moyen du nuage de point courant de la caméra.

entrée : $\mathcal{P}$ Le nuage de point provenant de la caméra
entrée : D La distance à garder par rapport aux murs
sortie : po la position objectif de la caméra
sortie : $\mathbf{u}$ le vecteur d'orientation de la caméra

1 $\mathcal{S} \leftarrow$ Downsampling($\mathcal{P}$)
2 $\mathcal{S} \leftarrow$ FiltrageDuSol($\mathcal{S}$)
3 $\mathcal{S} \leftarrow$ DécoupeAvant($\mathcal{S}$)
4 $\mathbf{p}^c \leftarrow$ Moyenne($\mathcal{S}$)
5 $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3; \lambda_1, \lambda_2, \lambda_3] \leftarrow$ AnalyseComposantesPrincipales($\mathcal{S}$)
6 $\tilde{\mathbf{u}} \leftarrow \mathbf{v}_3 - (\mathbf{v}_3 \cdot \mathbf{z}_c)\mathbf{v}_3$; `// Projection vers le plan` $\mathbf{x_c y_c}$
7 $\mathbf{u} \leftarrow \tilde{\mathbf{u}}\ \text{sign}(\tilde{\mathbf{u}} \cdot \overrightarrow{O_c \overline{p}^c}); \mathbf{u} \leftarrow \mathbf{u}/\|\mathbf{u}\|$
8 $\mathbf{r} \leftarrow \mathbf{z}_c \times \mathbf{u}$
9 `// Vérification de la largeur de` $\mathcal{S}$
10 **if** $y^c_{max} - y^c_{min} > w$ **then**
11 $\quad \alpha = \frac{y^c_{max} - y^c_{min}}{6}$; `// Calcul de la longueur de pas à prendre`
12 $\quad po^c \leftarrow p^c - D\mathbf{u} + \alpha\mathbf{r}$
13 **else**
14 $\quad po^c \leftarrow p^c + D\mathbf{r}$
15 **end**
16 **return** $po^c, \mathbf{u}$

Étant donné un nuage de points P provenant de la caméra, nous filtrons d'abord le sol en éliminant tous les points sous une certaine hauteur $z^g = s$. Ensuite à la ligne 3, nous découpons P pour obtenir un sous-ensemble S adjacent à la prochaine section à inspecter, c'est-à-dire que S est une partie (nous choisissons 1/3) de P vers la gauche de la caméra. Plus formellement, S est choisi tel que les coordonnées en y^c des points qui le composent satisfont

$$y^c_{max} - \frac{y^c_{max} - y^c_{min}}{3} \leq y^c \leq y^c_{max} \tag{3.1}$$

où y^c_{max} et y^c_{min} sont le maximum et le minimum des coordonnées y^c de $\mathcal{P}$.

La 5$^{\text{ième}}$ étape est de déterminer la normale $\mathbf{n}$ de la surface. Rusu (2009) explique que ceci peut-être fait en résolvant un problème des moindres carrés pour ajuster le modèle d'un plan

Π au nuage $\mathcal{S}$. On peut représenter le plan par un point $\mathbf{x}$ et un vecteur normal $\mathbf{n}$, et la distance d'un point $\mathbf{p}_i \in \mathcal{S}$ au plan est défini par $d_i = (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}$. On choisi $\mathbf{x} = \mathbf{p}^c$ la moyenne des points de $\mathcal{S}$.

$$\mathbf{x} = \mathbf{p}^c = \frac{1}{k} \cdot \sum_{i=1}^{k} \mathbf{p}_i \tag{3.2}$$

Suivant une formulation par moindres carrés, on cherche donc à mettre les $d_i = 0$. La solution de $\mathbf{n}$ peut être trouvée par Analyse des Composantes Principales (ACP) en calculant les vecteurs propres et les valeurs propres de la matrice de covariance $\mathcal{C} \in \mathbb{R}^{3\times3}$ de $\mathcal{S}$.

$$\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{p}_i - \mathbf{p}^c)(\mathbf{p}_i - \mathbf{p}^c)^\top, \mathcal{C}\mathbf{v}_j = \lambda_j \mathbf{v}_j, j \in \{0, 1, 2\} \tag{3.3}$$

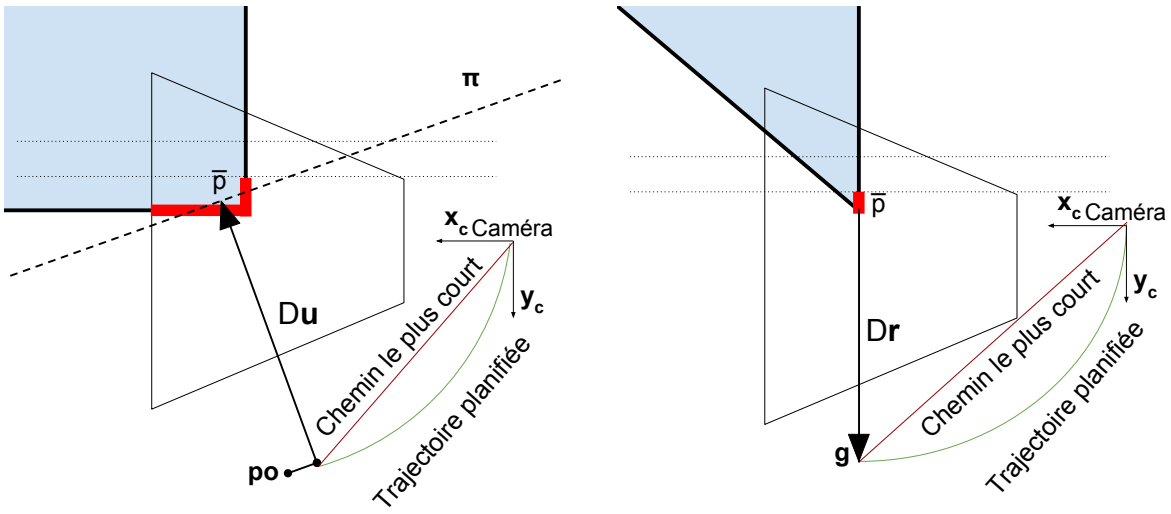
où $\mathcal{C}$ est semi-définie positive et les valeurs propres $\lambda_j \in \mathbb{R}^+$. Les vecteurs propres $\mathbf{v}_j$ forment ainsi une base orthonormale correspondant aux composantes principales de $\mathcal{S}$. En classant les valeurs propres par $0 \leq \lambda_3 \leq \lambda_2 \leq \lambda_1$, nous avons $\mathbf{v}_3$ le vecteur propre correspondant à la plus petite valeur propre λ_3 qui est donc une approximation de la normale $\pm\mathbf{n} \sim \mathbf{v}_3$ du plan Π, mais avec une ambiguïté de signe.

Le vecteur normal $\mathbf{v}_3$ est projeté sur le plan $\mathbf{x}_c\mathbf{y}_c$ sur lequel repose la caméra et pour résoudre l'ambiguïté de signe possible provenant de $\mathbf{v}_3$, $\mathbf{u}$ est pris pour pointer dans la direction du vecteur $\overrightarrow{O_c\overline{p}^c}$, c'est-à-dire de la caméra au centre de $\mathcal{S}$. L'algorithme retourne aussi la position objectif *po* de la caméra, calculé à partir de $-D\mathbf{u}$ pour garder la bonne distance au mur et $\alpha\mathbf{r}$ pour faire avancer le robot autour des coins de la structure. La Figure 3.4 permet de mieux visualiser cet effet ; $\mathbf{r}$ repose sur le plan Π et permet à la caméra de voir autour des coins pour passer au prochain mur à inspecter. Une limite de cette approche est que dans le cas de coins aigus ou de murs particulièrement minces, la caméra ne pourra avancer assez pour voir la suite du mur. Nous pouvons détecter ce cas quand la largeur de $\mathcal{S}$ tombe en dessous d'un certain seuil w ; c'est à ce moment que *po* est calculé par $po \leftarrow p^c + D\mathbf{r}$, forçant ainsi l'UGV à tourner autour du coin.

Finalement po^c est transformé dans le repère G pour fins de navigation. Le but du robot devient donc de bouger de façon à placer O_c à po^g et d'orienter la caméra selon $\mathbf{u}$.

3.2.2 Planification de trajectoire locale

Puisque notre robot est terrestre, la tâche de planification de trajectoire se simplifie en deux dimensions. Pour ce faire, le modèle 3D en construction est projeté sur une grille 2D pour



(a) (b)

Figure 3.4 Illustration du calcul de la prochaine pose objectif autour d'un coin a) dans le cas normal b) dans le cas d'un coin à angle aigu.

établir une carte d'occupation. On prend un sous-ensemble local de la carte de taille $2D \times 2D$ (nous rappelons que D est la distance à la structure désirée) autour de la position de la caméra pour calculer une trajectoire par champ de potentiel (Khatib, 1990; Choset *et al.*, 2005). Soit k cellules occupées aux coordonnées $\{x_j\}_{j=1}^{k}$. Nous avons la fonction de potentiel

$$N(x) = \alpha \|x - po^g\|^2 + \sum_{j=1}^{k} I_j(x) d_j(x) \tag{3.4}$$

où α est un scalaire. La fonction de répulsion d_j définie par :

$$d_j = \frac{1}{\beta \|x - x_j\|} \tag{3.5}$$

où β est un scalaire et $I_j(x)$ est le terme limitant la portée de d_j par

$$I_j(x) = \begin{cases} 1, & \text{si } \|x - x_j\| \leq D \\ 0, & \text{autrement} \end{cases} \tag{3.6}$$

Pour guider le robot à sa destination, il suffit de suivre le gradient négatif du champ de potentiel en envoyant des commandes de vitesses à l'UGV, i.e, $\dot{x} = -\nabla N(x)$ avec

$$-\nabla N(x) = 2\alpha(po^g - x) + \sum_{i \in J_x} \frac{1}{\beta \|x - x_i\|^3}(x - x_i) \tag{3.7}$$

où $J_x = \{j : I_j(x) = 1\}$ l'ensemble des cellules occupées de la carte dans un voisinage D de x.

En d'autres mots, considérons $\mathcal{M}_D = \{x : J_x \neq \emptyset\}$ la région à l'intérieur d'une distance D de la structure, pour une petite valeur de β le terme de répulsion dans l'équation 3.7 repousse le véhicule. Quand l'UGV se situe en dehors de $\mathcal{M}_D$, le premier terme de l'équation 3.7 devient dominant et attire le véhicule vers po^g. Nous arrêtons de suivre le gradient une fois que l'UGV se retrouve dans un certain rayon d'acceptation de la position objectif, après quoi nous recommençons l'algorithme 1.

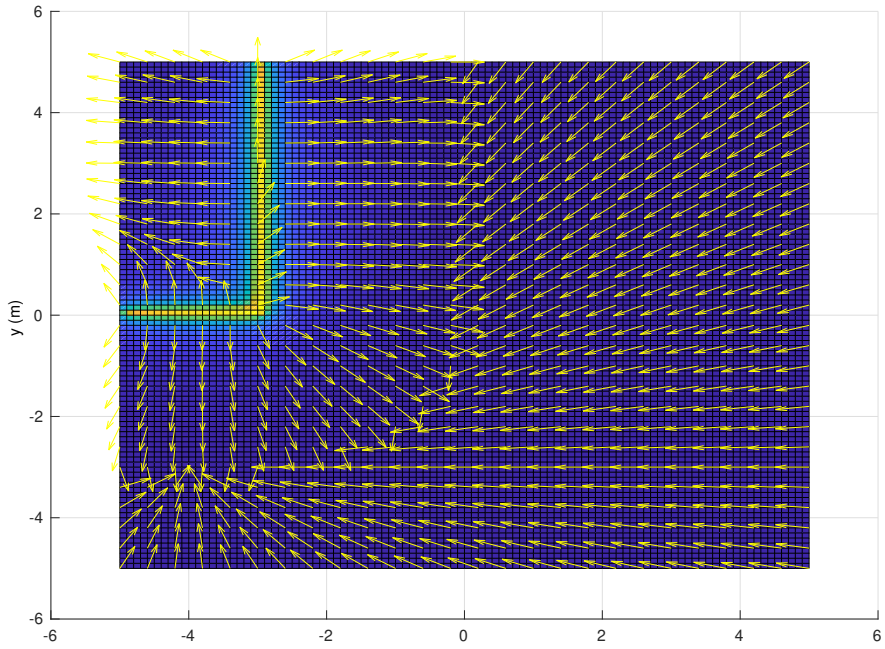
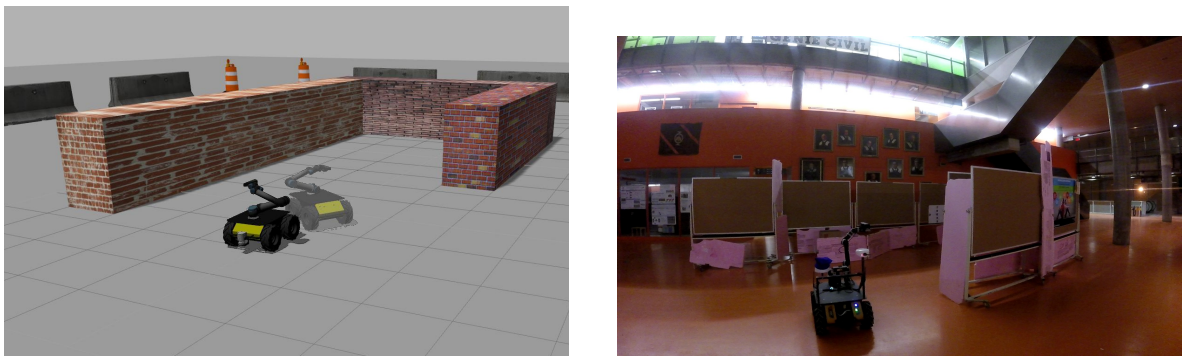


Figure 3.5 Exemple de visualisation du champ de potentiel. Les carrés jaunes représentent un mur et les vecteurs représentent la direction du gradient.

En somme, les équations et les algorithmes ci-dessus permettent au robot parcourir incrémentalement la surface de la structure en gardant un point de vue orthogonal aux murs.

3.2.3 Saut de cavités

Prenons maintenant l'exemple de la Figure 3.6, alors que le robot s'apprête à tourner autour du coin pour entrer dans la cavité. Le détecteur d'obstacles se déclenche en voyant le mur arrière. D'une part, nous voulons éviter d'entrer dans la cavité et d'autre part nous voulons empêcher le robot de s'approcher à moins d'une distance D d'un mur qui se retrouverait hors de son champ de vision. C'est pourquoi lorsque l'obstacle est détecté, le robot s'arrête et tourne le capteur de profondeur dans la direction de la détection. L'algorithme 1 est réexécuté avec le nouveau nuage de points S pris en avant du robot et la trajectoire d'inspection se poursuit vers la gauche.



(a) (b)

Figure 3.6 Exemple de détection d'obstacle et de saut de cavité. a) Le robot (transparent) tente de tourner autour du coin, mais détecte un mur vers l'avant. La trajectoire est replanifiée pour continuer vers la gauche sans entrer dans la cavité.

3.2.4 Fin de l'exploration de périmètre

La phase EP se termine lorsqu'une fermeture de boucle globale est détectée par le module de SLAM. Ceci peut être difficile si la trajectoire a accumulé trop d'erreurs ou si l'environnement ne comporte pas de caractéristiques uniques permettant au système de SLAM de définitivement détecter le retour à un endroit connu. C'est pourquoi nous recommandons l'ajout d'un objet unique à la position de départ du robot pour assurer une bonne fermeture de boucle. Dans nos tests, ceci consistait en une pancarte colorée placée sur la surface de la structure telle que dans la Figure 3.7.

3.3 Finition du modèle

Au terme de la phase EP, le modèle de la structure produit par le module de SLAM demeure incomplet pour deux raisons possibles. La première est due à des occlusions présentes dans

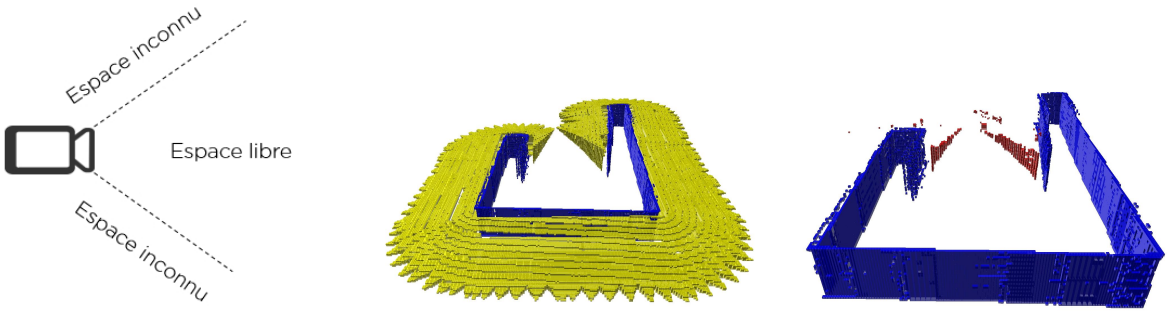
Figure 3.7 Exemple d'objet unique à placer au point de départ pour assurer une fermeture de boucle.

les régions explorées provenant de la structure elle-même par sa forme irrégulière ou par un échec du capteur de profondeur. La deuxième provient des régions non explorées qui ont été ignorées par la phase EP par la manœuvre expliquée à la section 3.2.3. Dans cette section, nous profitons du deuxième cas pour détecter les régions qui restent à explorer nous permettant ainsi compléter le modèle.

3.3.1 Détection des cavités

À mesure que le robot cartographie la structure, la séquence de nuages de points est rassemblée dans un grand modèle contenant tous les nuages que le module de SLAM considère comme utiles à intégrer. Suite à la fermeture de la boucle, nous avons une estimation optimale de toutes les poses du robot à partir desquelles chaque nuage de points a été capté. Grâce à cette information, il est possible d'effectuer un traçage de rayon dans une OctoMap (Hornung *et al.*, 2013) pour nous donner une représentation 3D de l'environnement qui inclut les espaces vide, occupé et inconnu.

Suivant la définition posée par Yamauchi (1997), un voxel frontière (*frontier*) est un voxel non occupé adjacent à un voxel de l'espace inconnu. Grâce à notre OctoMap, il est donc possible de calculer l'ensemble de tous les voxels frontières. Une particularité de l'utilisation d'une caméra de profondeur lors de l'inspection est que l'espace connu prend la forme d'un tronc de pyramide correspondant au champ de vision de la lentille. Ceci a pour effet, tel que l'on peut voir dans la Figure 3.8b que la carte finale inclut une grande quantité de voxels frontières correspondant aux hauts et aux bas des troncs de vue utilisés pour construire la carte.



(a) (b) (c)

Figure 3.8 Analyse de l'OctoMap avant le début de la phase d'exploration de cavités. a) Tronc de vue donnant lieu aux voxels frontières indésirables. b) En jaune les voxels frontières et en bleu les voxels faisant partie de la structure. c) En rouge les voxels frontières identifiant une entrée de cavité.

Algorithme 2 : Recherche de voxels frontières indiquant une entrée de cavité possible dans la OctoMap.

entrée : $\mathcal{G}$ Le graphe des poses optimisées de la carte et les nuages de points associés
sortie : $\mathcal{F}$ L'ensemble des voxels frontières

foreach $nuage^c$, $pose^g \in \mathcal{G}$ **do**
  $nuage^g \leftarrow T_c^g nuage^c$
  $nuage^g \leftarrow$ SeuillageDuSol($nuage^g$)
  $nuage^g \leftarrow$ SeuillageDuPlafond($nuage^g$)
  Insertion et traçage de rayon de $nuage^g$ dans l'OctoMap
end
foreach *voxel* $\in$ *OctoMap* **do**
  if *voxel.estVide() && 6-connexité inclut au moins un voxel d'espace inconnu* **then**
    Ajouter le voxel à l'ensemble des frontières candidates $\mathcal{C}$
  end
end
foreach $c \in \mathcal{C}$ **do**
  $v :=$ L'ensemble des voxels dans un voisinage k de c
  $\vec{n}_i^g \leftarrow$ ACP(v) ; `// Estimation de la normale par ACP`
  if $|\vec{n}_i{}_z^g| < \alpha$ && *voisinage* d_0 *de* c *n'est pas occupé* **then**
    Ajouter c à $\mathcal{F}$
  end
end
return $\mathcal{F}$

Suivant l'algorithme 2, nous pouvons filtrer les voxels frontières selon la normale de la surface composée des voxels dans un certain voisinage k du voxel considéré. Si $|n_z^g| < \alpha$ pour une certaine tolérance α le voxel est gardé. Une deuxième passe est réalisée pour enlever les fron-

tières trop proches de la structure ; ceci permet d'éliminer les défauts provenant d'occlusions ou d'échecs de perception. Le seuil d_0 utilisé peut être choisi par une fraction de la distance de sécurité, par exemple $d_0 = 0.1D$. Le résultat final de l'algorithme 2 appliqué à l'OctoMap présentée dans la Figure 3.8b est visible dans la Figure 3.8c.

Nous avons maintenant un ensemble de voxels faisant possiblement partie des entrées de cavités dans la structure. Ces voxels sont partitionnés par leur distance euclidienne selon un algorithme de remplissage par diffusion. Les partitions dont la surface est en dessous d'un certain seuil sont supprimées. Ce seuil peut être déterminé intuitivement par exemple en fonction de D et de la largeur du robot. Nous pourrions par exemple supprimer toute entrée ne donnant pas assez d'espace au robot pour manœuvrer. À la fin de cette procédure, chaque partition représente une entrée de cavité.

3.3.2 Exploration des cavités

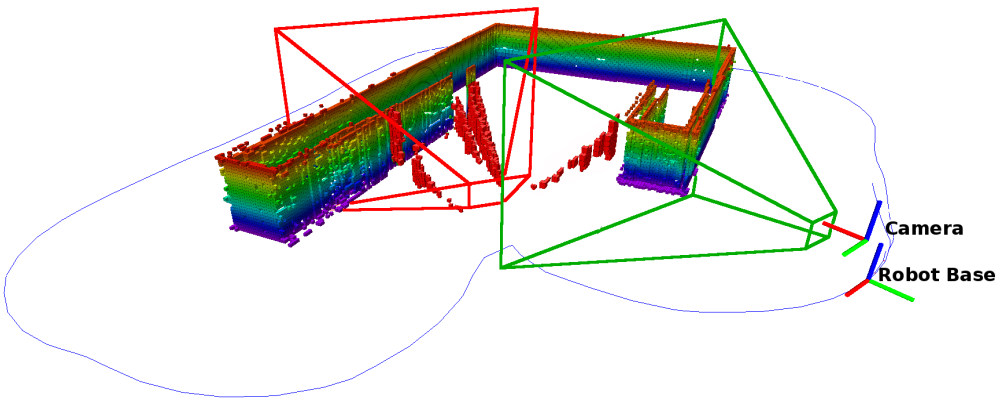


Figure 3.9 Début de la phase EC avec les points de vue associés à chaque entrée détectée.

Une fois les entrées de cavités trouvées, il est possible de commencer la phase d'exploration des cavités (EC). Pour ce faire, chaque cavité est explorée de façon similaire à la phase EP, c'est-à-dire par suivi de mur en gardant la caméra orthogonale à la surface, mais cette fois-ci à une distance $\Delta \in [\delta, D]$ où δ est la distance de perception minimale. Nous verrons plus tard comment choisir Δ, mais avant nous devons examiner la façon de choisir un point de vue (PDV) de départ pour débuter la phase EC.

En ordonnant les poses du graphe $\mathcal{G}$ provenant du système de SLAM par horodatage, nous choisissons pour chaque entrée la première pose qui possède une vue non obstruée au centroïde des voxels composant l'entrée. Cette liste de poses objectifs est à son tour ordonnée par horodatage et explorée en ordre. Comme nous pouvons l'observer dans la Figure 3.9, la détection d'entrée calculera typiquement deux entrées par cavité, parfois plus. Ceci provient

du fait qu'une grande cavité aura des voxels frontières de part et d'autre de l'entrée tel que l'on peut voir dans la Figure 3.8c. Pour éviter la redondance dans la phase EC, si le centroïde d'une entrée entre en vue lors de l'exploration, l'entrée est enlevée de la liste.

La phase EC requiert une méthode de navigation légèrement différente de la phase EP. Premièrement, la détection d'obstacles ne se fait maintenant que vers l'avant à une distance Δ du robot pour éviter de déclencher un virage prématuré. Nous modifions donc la ligne 12 de l'algorithme 1 à $po^c \leftarrow p^c - \Delta\mathbf{u} + \alpha\mathbf{r}$ et de même nous modifions la fonction du champ de potentiel pour générer des trajectoires à une distance Δ des murs. Nous pouvons détecter la fin de l'exploration d'une cavité lorsqu'une nouvelle fermeture de boucle est détectée, car les images prises à la sortie d'une cavité correspondront à des endroits déjà visités par le module de SLAM.

3.4 Résultats

3.4.1 Résultats en simulation

Le système a été implémenté en simulation en combinant du code en C++ et en Python à l'aide de le logiciel middleware Robot Operating System [1] (ROS). Les opérations sur les nuages de points ont étés implémentées grâce à la Point Cloud Library (PCL) de Rusu et Cousins (2011). Le système de SLAM utilisé est RTAB-Map (Labbé et Michaud, 2014), mais nous rappelons que tout autre système de SLAM 3D basé sur des graphes de poses est utilisable. L'une des particularités de RTAB-Map est qu'il supporte en entrée une source d'odométrie arbitraire, nous optons donc pour l'utilisation de l'odométrie provenant des roues du Husky. L'implémentation de nos algorithmes de mouvement incluant le choix de la prochaine pose et le calcul du champ de potentiel ont été faits dans la pile de navigation ROS (Marder-Eppstein *et al.*, 2010) que nous avons modifiée.

Le véhicule modélisé est présenté à la Figure 3.10 et consiste en un Husky A200 de Clearpath sur lequel un bras manipulater UR5 de Universal Robotics est installé. Bien que le UR5 possède 6 degrés de liberté, nous contraignons son mouvement à 1 seul joint pour respecter la prémisse du problème expliqué dans la section 3.1. L'organe terminal du manipulateur est remplacé par un capteur de profondeur générique aux paramètres de profondeur configurables et la simulation se déroule dans l'environnement de simulation Gazebo. Pour fins d'illustration, l'environnement d'essai est composé d'une structure formée de murets droits. La structure illustrée dans les figures précédentes est dénommée ci-après «Petit Γ» et le modèle «Grand Γ» est de la même forme, mais aux dimensions deux fois plus grandes. Notre

1. http ://www.ros.org/

Figure 3.10 L'UGV ClearPath Husky avec un modèle de Microsoft Kinect v1 à l'extrémité d'un manipulateur UR5 et un lidar balayeur vers l'avant.

algorithme est aussi comparé à l'algorithme classique d'exploration par frontières proposée par Yamauchi (1997) dans un scénario d'inspection de l'extérieur d'une maison et du modèle Petit Γ.

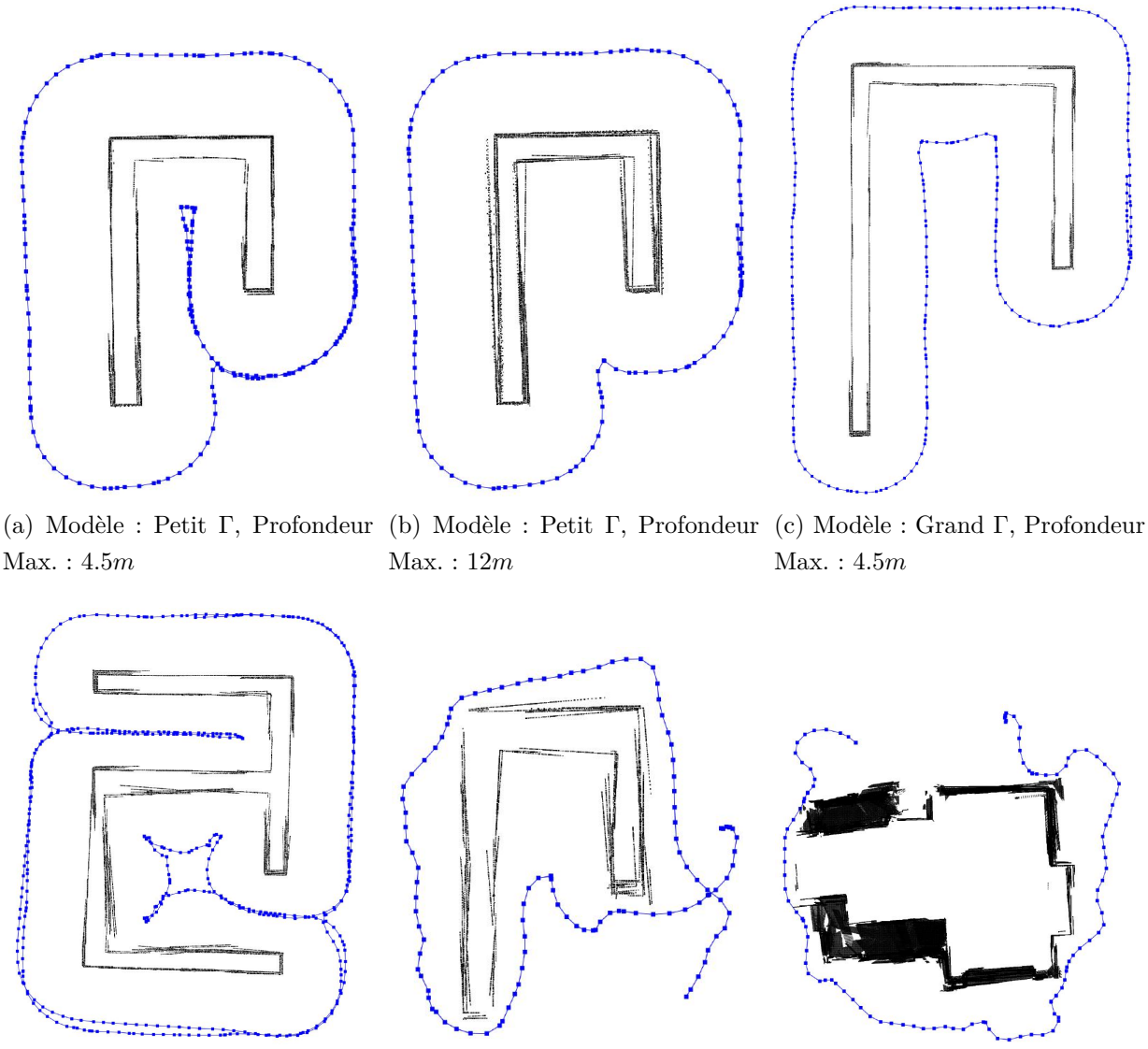
Variation de la taille de structure et de plage de détection profondeur

Les capteurs de profondeur ont la possibilité de configurer la limite de perception de profondeur ce qui, en plus de la taille de la structure, modifie la trajectoire suivie par l'UGV. Dans toutes les figures, le robot commence sa mission à droite de la structure et bouge initialement vers le bas. Dans la Figure 3.11a nous pouvons observer le comportement attendu en temps normal de l'algorithme. L'UGV fait un tour de structure pour fermer la boucle pour ensuite entrer dans la cavité. L'inspection se termine lorsque l'UGV ressort de la cavité. Dans la Figure 3.11b, la perception est augmentée à 12 mètres. Ici, on ne fait qu'un seul tour de la structure, car quand l'UGV tente de tourner le premier coin, il arrive à percevoir l'intérieur de la cavité. La phase de détection de cavité ne retourne donc rien et la mission se termine. Dans la Figure 3.11c, le mur de gauche est assez éloigné pour que le détecteur d'obstacles ne se déclenche. Ainsi, l'UGV entre dans la cavité dès la première passe et la mission se termine après un seul tour de la structure. Finalement, la Figure 3.11d démontre un scénario contenant plusieurs cavités et nous pouvons voir que l'algorithme réussit encore à cartographier la structure. Une première passe est réalisée en sautant les cavités et lors de la deuxième passe l'UGV s'arrête en haut à gauche après être sorti de la deuxième cavité. Le Tableau 3.2 présente quelques statistiques sur la longueur de trajectoire finale dans les différents scénarios

décrits.

Tableau 3.2 Résultats de simulation pour différentes tailles de la structure et profondeurs de caméra

Modèle	Taille du périmètre (m)	Profondeur de la caméra (m)	Longueur de la trajectoire (m)
Petit Γ	42	4.5	72.08
Petit Γ	42	12.0	53.79
Grand Γ	84	4.5	106.23
Grand a	94	4.5	179.65



(a) Modèle : Petit Γ, Profondeur Max. : 4.5m

(b) Modèle : Petit Γ, Profondeur Max. : 12m

(c) Modèle : Grand Γ, Profondeur Max. : 4.5m

(d) Modèle : Grand a, Profondeur Max. : 4.5m

(e) Petit Γ, Exploration par frontières

(f) Maison, Exploration par frontières

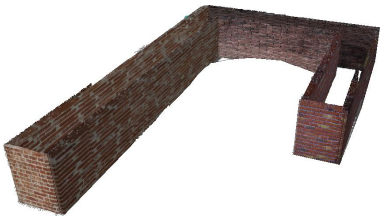
Figure 3.11 Projection du modèle reconstruit sur le plan $\mathbf{x}_g\mathbf{y}_g$ en noir et la trajectoire empruntée par le robot en bleu.

Comparaison à l'exploration par frontières

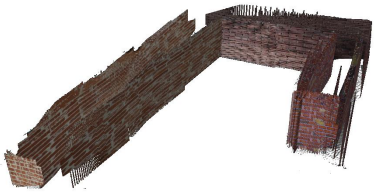
Nous comparons notre algorithme à une implémentation de l'exploration par frontières [2] (EF). L'algorithme prend en entrée les données d'un lidar balayeur 2D pour construire une carte d'occupation sur laquelle les frontières sont calculées. De plus, les bornes d'un polygone 2D

2. `http://wiki.ros.org/frontier_exploration`

délimitant la région à inspecter doivent être fournies. L'inspection se termine lorsqu'aucune frontière n'est trouvé à l'intérieur du polygone. Il existe trois différences majeures entre la solution que nous proposons et l'EF : 1) nous ne requérons pas d'information *a priori* à propos des limites de la région à inspecter ; 2) l'EF ne tente pas de se tenir à une distance fixe de la structure pour maximiser la portion de la surface cartographiée ; 3) La trajectoire de l'EF est parfois difficile à prévoir comparativement à une stratégie de suivi de mur, ce qui est important d'un point de vue de la convivialité. De plus, l'EF peut être affecté par le bruit des capteurs qui cause de légères différences dans la carte d'occupation produite ce qui cause ensuite une génération de trajectoires différentes à chaque exécution de l'algorithme. En revanche, notre algorithme produit une trajectoire similaire à chaque exécution. De plus l'EF ne prend pas en compte une distance de sécurité D par rapport à la structure ce qui cause parfois le calcul d'une position objectif très proche de la structure. Ceci est problématique, car, en plus de nuire au capteur de profondeur, l'UGV tend à rester coincé ou en oscillation sur place.



(a) Notre algorithme



(b) Exploration par frontier

(c) Exploration par frontier

Figure 3.12 Comparaison entre les résultats de notre algorithme par rapport à l'exploration par frontières avec un capteur de profondeur avec une portée de 4.5 m. On peut voir en (a) par rapport à (B) que l'EF n'assure pas une couverture complète de la surface visible du modèle.

Afin de comparer la couverture du modèle, nous utilisons le logiciel CloudCompare pour

calculer la projection du modèle reconstruit où $\mathcal{C}$ est aligné au modèle de référence discrétisé $\mathcal{C}_R$. Pour les deux modèles Petit Γ et celui de la maison, la discrétisation se fait avec une distance minimale de 0.1 m entre chaque point et seulement sur la portion allant jusqu'à la hauteur maximale perceptible H_{max}. Pour l'alignement de $\mathcal{C}$ à $\mathcal{C}_R$, une procédure de *iterative closest point* (ICP) (Rusinkiewicz et Levoy, 2001) a été utilisée. Ensuite, pour chaque point de $\mathcal{C}$, on recherche le point de référence le plus proche dans $\mathcal{C}_R$. Dans le cas où un point de $\mathcal{C}_R$ est associé à plusieurs points de $\mathcal{C}$, nous enlevons les points dupliqués pour obtenir l'ensemble des points uniques les plus proches de la structure. Le nombre de points dans cet ensemble devient notre métrique par laquelle nous estimons la couverture du modèle. Finalement, nous calculons aussi l'erreur moyenne qui correspond à la distance moyenne entre les points de l'ensemble unique et ceux du modèle.

Le Tableau 3.3 montre la différence entre notre stratégie et l'EF avec une caméra de profondeur limitée à 4.5 m. En somme, notre algorithme réussit systématiquement à obtenir une couverture plus élevée en plus d'une trajectoire plus courte. De plus, nous notons que la trajectoire lisse gardant toujours en vue la structure permet d'obtenir une erreur de reconstruction plus petite. De plus, la Figure 3.12 démontre qualitativement l'amélioration en termes des couvertures de notre algorithme et les lacunes au niveau de la reconstruction de l'EF.

Tableau 3.3 Comparaison entre notre algorithme et l'EF

		Notre algorithme	EF
Petit Γ	Trajectoire (m)	72.08	49.78
	# de points uniques les plus proches (maximum de 6116)	6063	5398
	Erreur moyenne (m)	0.05	0.12
Maison	Trajectoire (m)	59.89	47.55
	# de points uniques les plus proches (maximum de 10 889)	9182	7402
	Erreur moyenne (m)	0.05	0.12

3.4.2 Résultats expérimentaux

L'implémentation sur un vrai robot a été réalisée sous des conditions similaires à la simulation où un Husky est équipé d'un bras robotique Kinova Mico au bout duquel un capteur de profondeur Microsoft Kinect v2 est attaché. En guise de détecteur d'obstacle, un scanner lidar

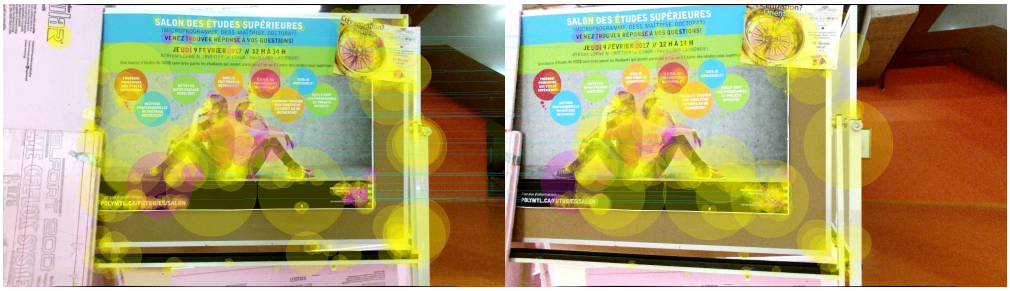
SICK LMS511 est installé à l'avant. Contrairement à nos simulations, nous effectuons nos tests dans un environnement intérieur dépourvu de signal GPS. C'est pourquoi ici l'odométrie provient de la fusion de seulement l'odométrie des roues et de la centrale inertielle dans l'EKF de (Moore et Stouch, 2014). Cette odométrie est ensuite envoyée à RTAB-Map (Labbé et Michaud, 2014) pour fins de localisation et de cartographie. L'intégralité des calculs se déroule sur un ordinateur à base d'un processeur Intel i5 sans l'aide d'un GPU. La structure à inspecter imite le modèle Petit Γ avec les dimensions de 8.2 m × 4 m et a été construite avec un mélange de panneaux d'isolation et des panneaux de liège amovibles. Pour assurer une fermeture de boucle, nous plaçons une pancarte colorée à l'endroit de départ du UGV visible dans les Figures 3.13b et 3.14a. Nous voyons clairement la différence entre la situation 3.14b où la mise en correspondance échoue et la situation en 3.14a où les caractéristiques uniques de notre pancarte aide le système de SLAM.

Puisque l'espace dans notre aire de test était limité, nous réduisons la portée de notre détecteur d'obstacles à 1.8 m et à un cône de 10° en avant du Husky. La profondeur de la caméra a été limitée à 4.0 m et nous configurons la distance au mur $D = 1.5$ m. De plus, puisque nous savons que l'UGV ne bouge pas en $\mathbf{z}^g$ et que nous supposons que le terrain est plat, nous contraignons RTAB-Map à 3 degrés de liberté ($\mathbf{x}^g$, $\mathbf{y}^g$ et l'angle de lacet ψ).

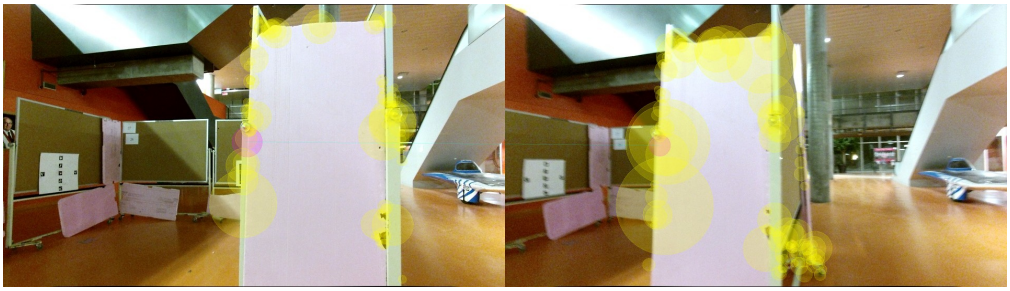
(a) (b)

Figure 3.13 (a) Le robot Husky utilisé dans nos expériences avec le bras articulé, le capteur de profondeur et le scanner lidar. Les autres capteurs non identifiés ne sont pas utilisés dans lors de l'expérience. (b) La structure intérieure sous inspection.

Le comportement de notre algorithme est illustré dans la Figure 3.15 et reflète presque le comportement prédit dans la simulation. Rappelons que pour une cavité, l'algorithme de



(a)



(b)

Figure 3.14 Tentatives de fermetures de boucle du système de SLAM RTAB-Map. Les lignes cyan représentent des mises en correspondance de caractéristiques entre l'image de référence (gauche) et l'image courante (droite). (a) Succès de la fermeture de boucle avec plusieurs mises en correspondance. (b) Tentative échouée de fermeture de boucle avec une seule mise en correspondance.

détection d'entrée retournera typiquement deux entrées de part et d'autre de l'ouverture dans la structure. En temps normal, la deuxième entrée sera enlevée de la liste des candidats lorsque son centroïde entrera en vue de la caméra de profondeur à la sortie de la cavité. Il semblerait que lors de l'expérience, la détection du centroïde a échoué, car l'UGV s'est retrouvé entre la structure et le centroïde à détecter. Ceci cause l'UGV à effectuer un deuxième tour autour de la structure avant de s'arrêter près de l'entrée. Nous voyons donc que si une partie de l'algorithme échoue, dans le pire des cas le robot effectue simplement un deuxième tour.

Performance à l'exécution

Le robot réussit à faire l'exploration du périmètre en sautant la cavité interne, le tout en temps réel malgré l'exécution du SLAM 3D et des différents algorithmes d'estimation d'odométrie et de planification de trajectoire. La partie la plus lente de l'algorithme reste la construction et l'analyse de l'OctoMap pour la détection des cavités qui peut prendre plusieurs secondes de temps. Bien que la représentation multirésolution d'un OcTree nous permet d'accélérer les requêtes à la carte, un problème existe néanmoins dans l'insertion des points et du tra-

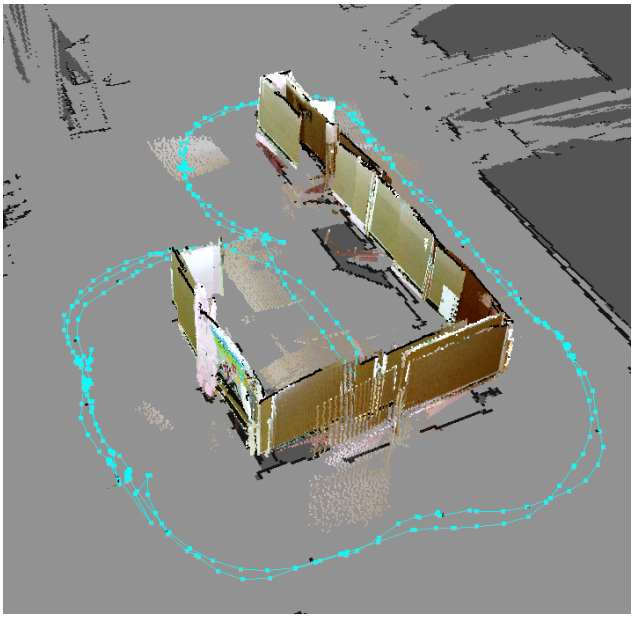


Figure 3.15 Résultat expérimental, en cyan la trajectoire, en gris la carte d'occupation et d'obstacles créée par l'assemblage des balayages du lidar et en couleur la reconstruction du modèle par nuage de points.

çage des rayons associés qui se fait séquentiellement sans l'utilisation de fils d'exécution en parallèle. Ceci pourrait être réglé en changeant la représentation utilisée d'OctoMap à une représentation parallélisable tel que la SkiMap récemment proposé par Gregorio et Stefano (2017).

3.4.3 Sources de bruit et différences entre la simulation et l'expérience réelle

En effectuant nos expériences, nous notons plusieurs différences entre notre environnement de simulation et la réalité. La première que nous remarquons est la quantité de bruit dans les capteurs. Dans notre environnement Gazebo, la caméra de profondeur est simulée avec un bruit gaussien sur les mesures de distances et chaque pixel de l'image de profondeur est en fait simulé par les mêmes mécanismes de simulation de laser. En réalité, les capteurs fonctionnant par ToF possèdent des caractéristiques de bruit légèrement différentes, entre autres, du bruit poivre et sel, et du bruit de chatoiement (des petites taches dans l'image). Cette dernière est particulièrement importante, car elle peut facilement faire échouer le calcul de la prochaine pose objectif tel que présenté dans la Figure 3.16.

En traitement d'images, le bruit poivre et sel serait normalement éliminé par un filtre médian (Jayaraman, 2009). Par contre, puisque nous voulons utiliser les images de profondeur pour la reconstruction et la navigation de notre robot, nous voulons préserver autant que possible les

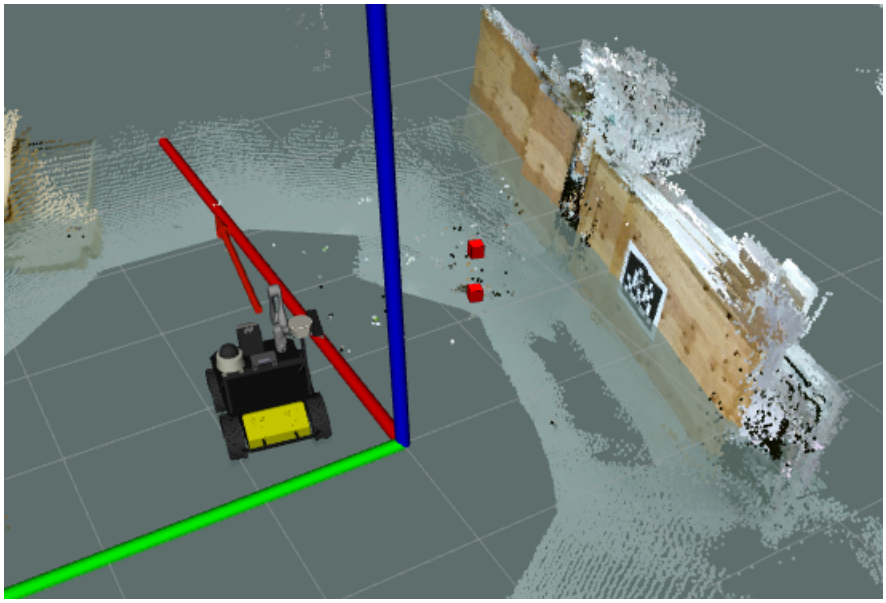


Figure 3.16 Échec du suivi de la structure par la présence de bruit de chatoiement. Les cubes rouges sont en fait des tâches qui ont été considérées comme une partie de la structure alors qu'elles n'existaient pas en réalité.

détails originaux des données. Le bruit poivre et sel est donc éliminé simplement par seuillage en imposant une limite supérieure et inférieure sur les distances perçues. Tout point en dehors de la plage acceptable n'est pas projeté dans le nuage de points utilisé pour la navigation et la cartographie. Pour cette même raison, nous n'appliquons pas de filtre bilatéral sur l'image de profondeur pour éliminer le bruit de chatoiement, car ce filtre pourrait lisser les contours de certains obstacles. À la place, nous appliquons un filtre de déchatoiement qui cherche des groupes de pixels connectés et qui impose ensuite une taille minimale sur ces groupes. Ceci est fait directement sur l'image de profondeur avant la projection à un nuage de points. Pour calibrer les paramètres du filtre, nous plaçons l'UGV à l'endroit où le plus de bruit est détecté et nous augmentons progressivement la taille maximale des taches à enlever jusqu'à ce que l'image de profondeur soit libre de bruit. Il faut aussi calibrer la différence maximale entre deux pixels pour les considérer dans le même groupe. Ce dernier paramètre peut être deviné intuitivement puis raffiné expérimentalement en inspectant une scène contenant plusieurs objets à diverses distances.

Outre le bruit de capteur, certaines différences inattendues se sont présentées physiquement sur le robot, causant une dégradation de la performance de cartographie. Premièrement, en raison de la taille des panneaux et de la distance limitée autour de la structure, le bras Kinova a été étendu à une hauteur de 1.2 m causant ainsi des vibrations sur la caméra de profondeur et des images floues envoyées au système de SLAM. La fermeture de boucle échoue parfois

en raison d'une mauvaise extraction des caractéristiques de l'image. Ce problème pourrait être évité de plusieurs façons : en utilisant un meilleur contrôle sur l'accélération maximale du robot, en installant le capteur sur une plateforme orientable plus rigide et en opérant dans un environnement mieux illuminé pour réduire le temps d'exposition du capteur RGB. Deuxièmement, le modèle Petit Γ utilisé en simulation était un assemblage de plusieurs prismes rectangulaires créant ainsi une structure parfaitement lisse. Par contre, dans notre assemblage de panneaux de liège, il existait plusieurs trous et ouvertures dans la structure auxquels l'algorithme 1 pour le suivi de mur est sensible. En effet, lorsqu'elle est devant une fenêtre, la caméra de profondeur tend à capter des surfaces à l'intérieur de la structure affectant ainsi le calcul de la normale de la surface. Pour amenuiser ce problème, il est utile de limiter la portée du capteur de profondeur à une valeur légèrement au-dessus de la distance au mur D.

En somme, notre algorithme est surtout sujet à deux sortes de cas d'échecs : i) l'échec du calcul de la trajectoire d'inspection par la présence de trous et de fenêtres dans la structure et ii) l'échec de la fermeture de boucle globale que nous pouvons mitiger à l'aide d'un marqueur visuel unique.

3.5 Conclusion

En conclusion, nous avons présenté une méthode pour la génération de trajectoires permettant de guider un UGV équipé d'une caméra de profondeur mobile pour l'inspection d'une structure fermée inconnue. Notre méthode ne requiert aucune information *a priori* sur la taille ou sur la géométrie de la structure. Utilisé en conjonction avec un système de SLAM visuel performant, nous obtenons une grande couverture de la surface de la structure dans le modèle reconstruit malgré les limites de la plateforme. Le système proposé est validé en simulation et dans des tests expérimentaux. Par une comparaison à l'algorithme classique d'exploration par frontières, nous démontrons clairement le gain en performance dans le cas d'une structure réaliste telle qu'une maison.

Finalement, la méthode proposée favorise la couverture de la surface du périmètre de la structure par le capteur, mais elle ne garantit pas que l'entièreté de la surface sera cartographiée. Pour ce faire, nous pourrions ajouter un planificateur de trajectoire secondaire pour déplacer la caméra localement au moyen du bras articulé pour assurer une couverture complète à chaque point d'inspection. De plus, il reste plusieurs améliorations possibles à apporter à l'algorithme, notamment l'utilisation d'une représentation autre que l'OctoMap telle que AtomMap, SkiMap ou les TSDF ou du moins la parallélisation du calcul des entrées de cavité. D'un autre côté, nous pourrions profiter du bras articulé pour ajouter des degrés de liberté à

la caméra de profondeur. Ceci permettrait de faire une planification de trajectoire secondaire pour optimiser davantage le positionnement de la caméra ou même de faire une planification de trajectoire globale prenant en compte les possibilités additionnelles.

CHAPITRE 4 INSPECTION D'ÉOLIENNES PAR UAV

Dans ce chapitre nous abordons l'automatisation d'inspection d'éoliennes par un véhicule aérien autonome. Ce projet a été réalisé conjointement avec une compagnie privée opérant dans le secteur des drones commerciaux.

4.1 Description du problème et méthode d'inspection

Les éoliennes doivent régulièrement être inspectées pour détecter et réparer le plus tôt possible les failles structurelles qui peuvent apparaître avec le temps à cause des intempéries ou quand elles sont frappées par la foudre. La peinture recouvrant celles-ci étant moins élastique que le composite de fibre de verre dont elles sont construites, un défaut structurel se manifeste inévitablement à la surface de la pale sous la forme de fissures ou de trous. Une inspection visuelle régulière peut donc être réalisée pour déceler et réparer toute irrégularité dans la structure avant qu'un bris catastrophique ne se produise.

En temps normal, une inspection se fait par une équipe de deux personnes devant grimper l'éolienne. L'équipe attend d'abord que le vent fasse tourner le rotor jusqu'à ce que la pale à examiner pointe vers le sol. À ce moment, le frein est enclenché et l'angle d'inclinaison des pales est ajusté pour que le vent ne crée pas de couple sur le rotor. L'équipe grimpe ensuite la tour jusqu'au sommet, ce qui peut prendre une dizaine de minutes si l'ascenseur est non fonctionnel, chose qui arrive fréquemment. Une fois rendu au rotor, un frein mécanique secondaire est activé pour bloquer la rotation des pales. L'équipe fait ensuite une descente en rappel le long de la pale pour faire l'inspection visuelle et tactile de la surface. Une fois l'inspection terminée le processus est répétée pour les deux autres pales. Inspecter une éolienne peut prendre entre deux et quatre heures selon le nombre de bris à documenter.

Certains opérateurs d'UAV, offrent maintenant des services d'inspection d'éoliennes. Une caméra haute définition est montée sur un cardan stabilisateur à trois axes, lui-même monté sur le véhicule. Ce dernier est parfois aussi augmenté d'un télémètre permettant au pilote de savoir à quelle distance il est de la structure, chose difficile à estimer lorsque le véhicule est à 100 mètres d'altitude. Ces inspections peuvent prendre jusqu'à trois employés, dont un pilote, un opérateur pour la caméra et un observateur sous l'éolienne.

Le but du projet est d'explorer les solutions possibles pour réaliser un système capable de piloter un quadricoptère de façon autonome autour des pales d'une éolienne à l'aide de capteurs à coût raisonnable (étant donné les contraintes de charge utile) avec seulement un

pilote de sécurité de présent et le moins d'informations *a priori*. Ce projet peut aussi servir de tremplin vers un projet dans lequel aucun humain n'interviendrait, par exemple pour une flotte d'UAV chargée d'inspecter régulièrement un parc éolien entier. Puisque manœuvrer précisément autour des pales demanderait l'usage de capteurs dépassant le budget du projet et nécessiterait que l'UAV soit équipé de capteurs de position haute précision tel qu'un GPS différentiel, nous simplifions le problème en un problème à deux dimensions où le véhicule cherche à suivre le bord d'attaque (ou le bord de fuite) des pales. Deux approches ont été développées, l'une entièrement au moyen de scanners laser et l'autre par caméras stéréo.

La mission se divise en deux phases principales : l'approche du rotor et le suivi de ses pales qui peuvent être gérées par une simple machine à états finis. Pour l'approche du rotor, nous comparons une approche à longue distance au moyen de traitement d'images à une approche à tâtons où l'UAV remonte la tour au moyen de scanners lasers. Pour le suivi des pales, nous comparons encore une méthode par traitement d'images à une méthode de suivie à tâtons. La compagnie pour laquelle nous travaillons désirait que le système soit implémenté sur quadricoptère md4-1000. Nous installons sur le véhicule deux scanners lasers LeddarTech Vu8, l'un horizontal et l'autre vertical, ainsi qu'une caméra stéréo ZED. Il est à noter que ces capteurs ne servent qu'à la navigation de l'UAV alors que la caméra haute définition sur stabilisateur demeure le moyen par lequel prendre des photos des défauts à la surface des pales.

Figure 4.1 Systèmes de coordonnées sur le md4-1000

Tableau 4.1 Repères présents dans le système d'inspection d'éoliennes

Symbole	Nom	Description
W	*World*	Suivant la convention *East-North-Up* (ENU), centrée au point de décollage du véhicule avec l'axe x pointant vers l'Est, y pointant vers le Nord et z pointant vers le haut.
B	*Body*	Encore suivant la convention ENU, l'axe x pointant vers la droite, y vers l'avant et z vers le haut.
BT	*Body Tangent*	Repère centré sur, B mais compensé pour ses angles de roulis et de tangage (mais pas de lacet). En d'autres mots, le repère BT est donc tangent à la surface de la Terre et centré sur B.
C[L, R]	*Camera*	Suivant la convention de représentation d'images, nous avons l'axe x vers la droite, y vers le bas et z sortant de la caméra. Les suffixes L et R indiquent les caméras gauche et droite respectivement. Par souci de brièveté si le repère est écrit C sans préciser la caméra gauche ou la caméra droite, il indique la caméra gauche.
L[H, V]	*Laser*	Les lasers aussi suivent un repère main droite avec x vers l'avant, y à gauche et z vers le haut. Les données de distance du laser se situent toutes sur son plan XY. Les suffixes H et V indiquent la direction du balayage respectivement horizontal et vertical, par rapport au repère B.
T	*Turbine*	Le système de coordonnées de la turbine est centré sur le rotor avec x vers la droite (vu de devant), y vers l'intérieur de la nacelle et z vers le haut.

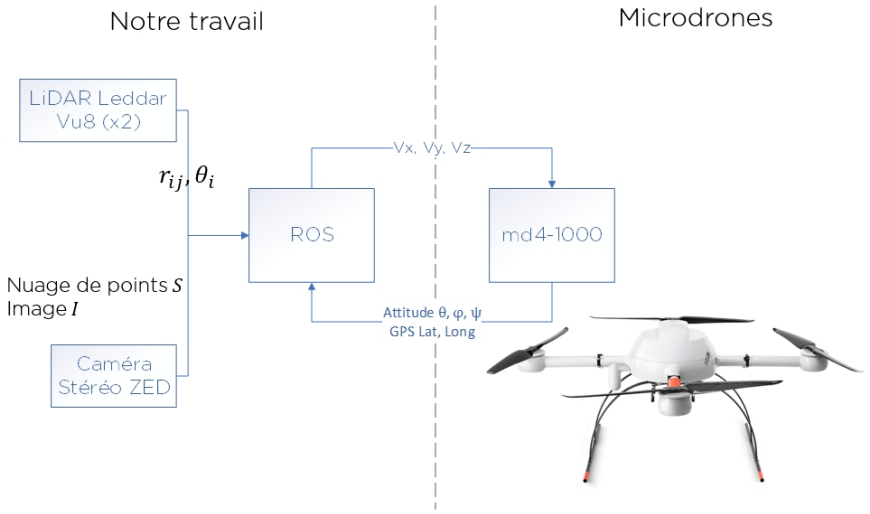


Figure 4.2 Vue d'ensemble du système développé

4.2 Approche du rotor

4.2.1 Approche de loin avec caméra

L'approche à longue distance comporte plusieurs avantages dont l'absence du besoin d'informations *a priori* à propos du placement des pales. Avec une vue d'ensemble de l'éolienne à inspecter, il devient possible de mesurer l'angle des pales, une information cruciale qui sera exploitée lorsque l'UAV devra choisir où se diriger lorsqu'il est près du rotor. Le deuxième avantage est qu'il est possible de mesurer l'angle de lacet relatif entre l'UAV et l'éolienne, permettant ainsi de placer le véhicule perpendiculairement à la surface des pales. Cette méthode requiert tout de même la présence d'un détecteur de proximité avec une portée d'au moins 10 mètres ; en revanche elle reste beaucoup moins dispendieuse que l'utilisation d'un scanner laser.

Dans cette section nous détaillons une méthode légèrement modifiée de celle de (Stokkeland *et al.*, 2015) pour détecter le centre du rotor d'une éolienne à partir d'une image. En somme, les étapes restent les mêmes, mais une robustesse additionnelle est introduite au moyen d'algorithmes de regroupement ; une représentation différente des pales détectées est utilisée et un filtre de Kalman non linéaire (au lieu d'un filtre de Kalman linéaire) est proposé.

La méthode de détection du rotor se divise en 6 étapes et repose principalement sur des raisonnements géométriques,

1. Prétraitement de l'image
2. Détection de contours
3. Détection de lignes
4. Recherche de la tour
5. Recherche des candidats de segments de pales
6. Procédure de filtrage des candidats et calcul de la position du rotor

Étape 1 : Prétraitement de l'image

Avant de commencer la détection, quelques corrections à l'image doivent être apportées. Puisque la procédure ne requiert pas de données de couleur, une conversion RGB en échelle de gris est effectuée au moyen de la transformée

$$Y \leftarrow 0.299R + 0.587G + 0.114B \tag{4.1}$$

où Y est le niveau de gris du pixel entre 0 et 255 ce qui donne comme résultat l'image de la Figure 4.3 (B). Les coefficients utilisés proviennent de la norme BT.601 de l'International Telecommunication Union (2011) pour le système d'encodage de couleurs YCbCr. En d'autres mots, nous effectuons une conversion de RGB à YCbCr mais nous ne gardons que le canal Y représentant la luminance. Pour éliminer la présence de bruit et atténuer les textures à haute fréquence pouvant donner lieu à une surdétection de lignes lors de la transformée de Hough, nous appliquons un filtre moyenneur 3×3 par convolution. Pour l'image floutée F, nous avons donc

$$F = \left(\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) * H \tag{4.2}$$

avec le résultat visible à la Figure 4.3 (C). Pour les convolutions sur les bords de l'image, nous répétons la valeur du pixel du bord pour les données manquantes dans l'opération de convolution. Pour augmenter le contraste et corriger les erreurs d'exposition de la caméra, une égalisation d'histogramme est appliquée. Pour l'image originale I et l'image égalisée H que l'on peut voir dans la Figure 4.3 (D), nous avons

$$p_n = \frac{\#\text{ de pixels d'intensité } n}{\#\text{ de pixels total}}, \quad n = 0, \ldots, 255 \tag{4.3}$$

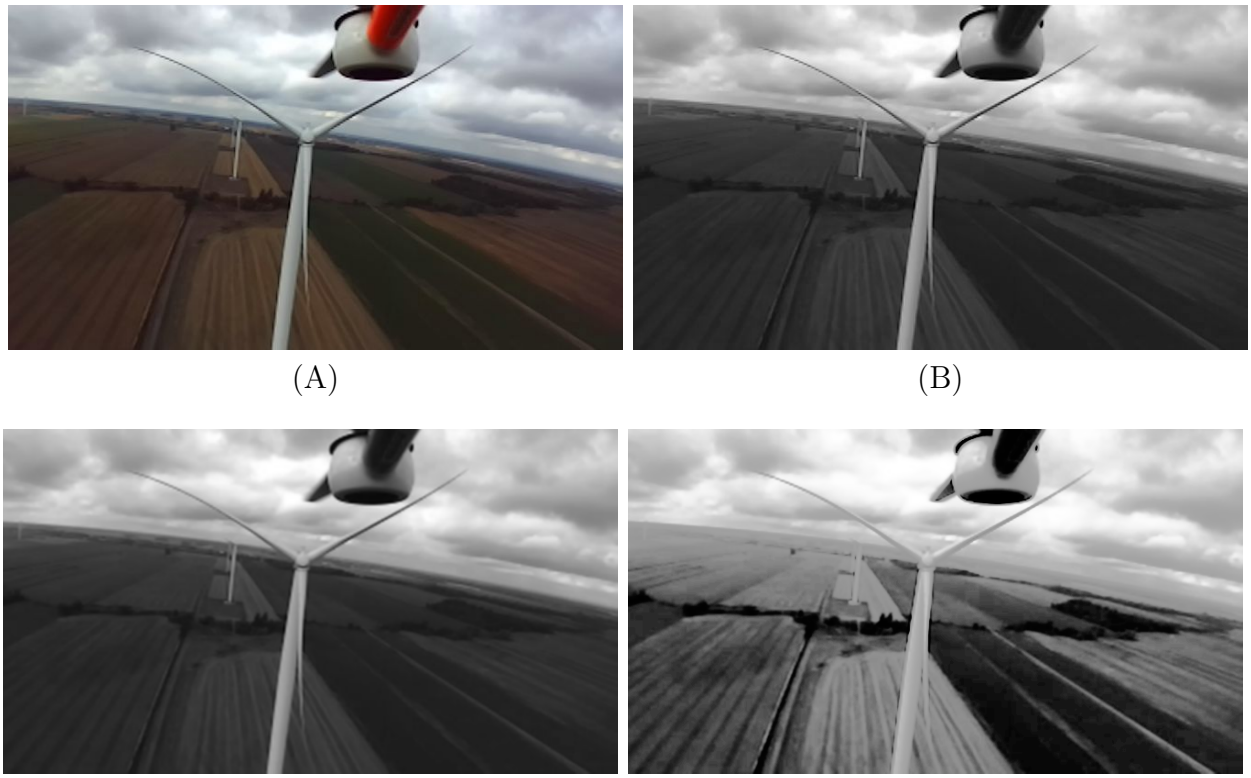
$$H_{i,j} = \text{floor}(255 \sum_{n=0}^{I_{i,j}} p_n) \tag{4.4}$$

où floor() est l'opérateur arrondissant à l'entier inférieur le plus proche.

Étape 2 : Détection de contours

La détection de contours se fait au moyen de l'algorithme de Canny (1986) et le résultat de la procédure est visible dans la Figure 4.4.

Outre la méthode de Canny, il existe des méthodes modernes plus performantes que celle de Canny, par exemple Xie et Tu (2015) proposent un réseau de neurones entièrement convolutionel capable de détecter beaucoup mieux les contours, et ce, sans avoir à manuellement ajuster ses paramètres. Par contre dans nos tests, la haute précision de la méthode retournait trop de contours, et il devenait difficile de passer aux prochaines étapes de la détection.



(A) (B)

(C) (D)

Figure 4.3 (A) Image originale (B) Image en niveaux de gris (C) Image floutée par un filtre moyenneur (D) Image avec égalisation d'histogramme.

Figure 4.4 Résultat de l'application du filtre de Canny avec $T_b = 27$ et $T_h = 81$.

Étape 3 : Détection de lignes

Une fois les contours trouvés, nous pouvons les utiliser pour détecter des segments de droite dans l'image. Dans le cas où des parties du véhicule sont visibles dans l'image, il suffit d'appliquer un masque avant de débuter cette étape. Nous appliquons la transformée de Hough probabiliste de Matas *et al.* (2000) à fin de détecter des segments de droites dans l'image de contours. Dans la Figure 4.5 tous les segments de couleur sont ceux détetectés par la transformée de Hough, la couleur reflète l'étape de sélection i.e. rouge pour l'étape 3, mauve pour l'étape 4, rose pour l'étape 5 et diverses couleurs pour chaque pale de l'étape 6.

Étape 4 : Recherche de la tour

Stokkeland explique que la partie la plus facile à détecter est la tour puisqu'elle est toujours verticale et répond fortement à la détection de lignes. Pour la trouver, il suffit de chercher une ligne verticale (compensée pour l'angle de roulis de l'UAV) dont l'un des sommets repose dans la base de l'image. Puisqu'il arrive parfois que la détection de ligne défasse la tour en plusieurs segments, une fois le segment initial choisi, on recherche aussi des prolongements de ce segment à une certaine distance du plus haut sommet.

Étape 5 : Recherche des pales

À la fin de l'étape 4, nous avons un ensemble de segments appartenant à la tour. À partir du plus haut point de ces segments, nous recherchons tout autre segment ayant un sommet dans un certain rayon du point. Ces segments sont considérés en tant que candidats à être des segments d'une pale.

Nous divergeons de la méthode proposée par Stokkeland où il tente de regrouper les segments selon l'angle par rapport à la tour au travers d'une procédure vorace simple où un nouveau groupe est créé lorsqu'un segment est à un angle au-dessus d'un certain seuil du groupe courant. Nous notons que ceci fonctionne acceptablement sur le jeu de données de Stokkeland où le ciel est clair et son éolienne est sur une colline près de la mer, mettant donc l'horizon bien en dessous du rotor. Lors de nos essais, tant en simulation que sur le terrain, nous avons noté que l'horizon était très proche du rotor, donnant lieu à une grande présence de segments près du rotor, faussant ainsi les résultats de la recherche de pales. C'est pourquoi nous implémentons l'étape 5 avec un algorithme de groupage formel qui inclut la résistance au bruit de mesure, nommé *Density-based spatial clustering of applications with noise* (DBSCAN) proposée par Ester *et al.* (1996).

La particularité de DBSCAN comparativement à d'autres algorithmes de partitionnement

tel que k-means est que nous n'avons pas à fournir le nombre k de partitions recherchées. À la place, nous ne fournissons que deux paramètres, ϵ (eps) la distance d'un point à son entourage, et MinPts le nombre minimum de points pour qu'une partition soit formée. Pour chaque segment, nous projetons leur angle par rapport à la tour sur un cercle unitaire. Ceci permet d'exécuter DBSCAN dans un espace euclidien pour regrouper les segments adjacents.

Une fois les groupes formés, Stokkeland utilise une stratégie de vote pour éliminer les fausses détections. La moyenne des angles est calculée et une procédure de vote débute s'il y a plus de deux groupes. Nous en choisissons deux au lieu de trois tel que proposé par Stokkeland pour prendre en compte le cas où une pale est vis-à-vis la tour. Sachant que les éoliennes ont toujours trois pales espacées de 120 degrés, chaque groupe calcule sa distance angulaire par rapport aux autres et vote pour les groupes n'étant pas à 120 degrés d'eux-mêmes. Dans le cas où DBSCAN ne ressort que deux groupes, la direction de la troisième pale est approximée à 120 degrés des deux autres pales.

Étape 6 : Recherche du rotor et filtrage

L'étape 5 nous a donc fourni trois groupes de segments représentant chacun les pales de l'éolienne. Le centre du rotor peut donc être calculé en moyennant chaque groupe puis en prenant la moyenne de l'intersection des prolongements de chaque segment.

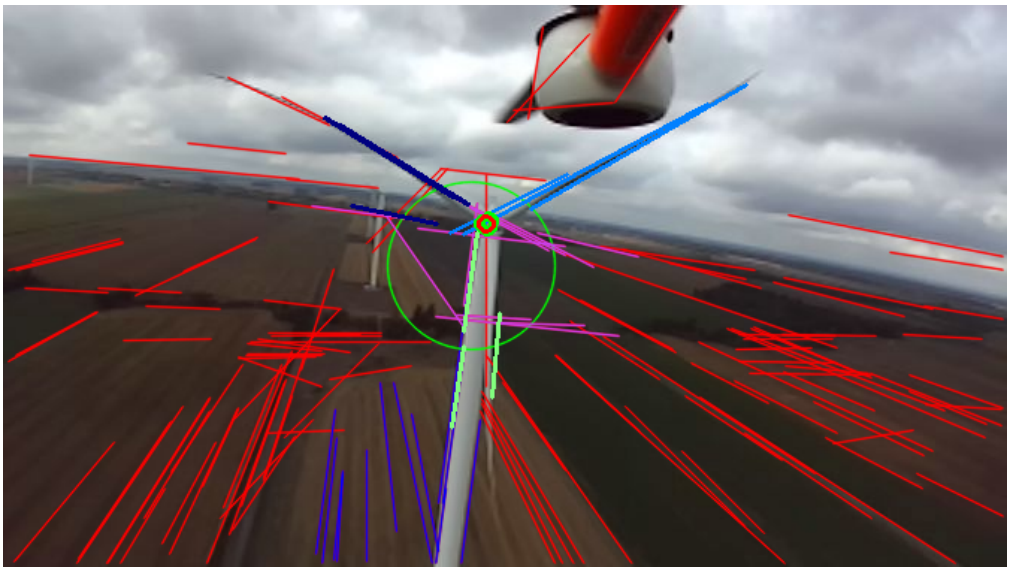


Figure 4.5 Résultat d'une détection réussie à la fin de la procédure.

Dans la Figure 4.5, nous pouvons observer les étapes 3 à 6 en opération. Toutes les lignes sont le résultat de la transformée de Hough appliquée à l'image des bordures suite à l'application

de l'algorithme de Canny. Au bas de l'image, les lignes mauves sont les candidates à être des lignes de la tour. Elles se font rejeter car elles ne s'étendent pas assez haut dans l'image. Le grand cercle vert est celui tracé au plus haut point de la tour dans lequel les segments de pales sont recherchés. Les segments roses sont les candidats de pales rejetés par l'algorithme de partitionnement DBSCAN et la procédure de vote. Les pales finales sont indiquées par les lignes bleu, bleu foncé et vert pâle. Finalement, avec l'application des étapes 5 et 6 on peut voir que l'intersection des pales se croise bien sur le centre du rotor.

Le point détecté passe ensuite dans un filtre de Kalman linéaire permettant ainsi de garder une estimation de la position du rotor dans l'image lorsque la détection échoue. Le vecteur d'état $\mathbf{x}_k$ contient la position et la vitesse de déplacement en coordonnées image du centre du rotor. La matrice de transition $\mathbf{F}_k$ contient les termes $\gamma_x, \gamma_y \in [0, 1]$ faisant décroître la vitesse exponentiellement avec le temps. Ceci empêche l'estimation de diverger à l'infini quand aucune mesure n'entre dans le filtre pendant de longues périodes de temps. Nous précisons que les équations suivantes sont en temps discret et le filtre suit la même fréquence que celle de la caméra.

$$\underbrace{\begin{bmatrix} u(k) \\ v(k) \\ \dot{u}(k) \\ \dot{v}(k) \end{bmatrix}}_{\boldsymbol{x}_k} = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & \gamma_x & 0 \\ 0 & 0 & 0 & \gamma_y \end{bmatrix}}_{\boldsymbol{F}_k} \underbrace{\begin{bmatrix} u(k-1) \\ v(k-1) \\ \dot{u}(k-1) \\ \dot{v}(k-1) \end{bmatrix}}_{\boldsymbol{x}_{k-1}} + \boldsymbol{w}_k \tag{4.5}$$

où $\boldsymbol{w}_k$ est un bruit blanc gaussien. Le modèle de mesure inclut seulement la détection du point dans l'image.

$$\underbrace{\begin{bmatrix} z_x \\ z_y \end{bmatrix}}_{\boldsymbol{z}_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\boldsymbol{H}_k} \underbrace{\begin{bmatrix} u(k) \\ v(k) \\ \dot{u}(k) \\ \dot{v}(k) \end{bmatrix}}_{\boldsymbol{x}_k} + \boldsymbol{v}_k \tag{4.6}$$

Le terme d'entrée de commande $\boldsymbol{u}_k$ pourrait être utilisé pour prendre en compte les vitesses angulaires commandées au robot. Par contre sachant qu'elles ne seront pas disponibles sur le lien de télémétrie du md4-1000, nous ne les prenons pas en compte. La meilleure façon d'améliorer le filtre serait d'ajouter des mesures de $\dot{u}$ et de $\dot{v}$, chose qui pourrait être faite par des méthodes de calcul de flux optique.

Une fois le point (u, v) trouvé dans l'image, il suffit d'appliquer le modèle de caméra sténopé

de la section 2.6.1 et sa matrice de paramètres intrinsèques K pour obtenir un vecteur unitaire u_C dans le repère caméra pointant vers le rotor. Puisque la distance entre l'UAV et le rotor est beaucoup plus grande que la distance entre la caméra (C) et le centre du véhicule (B) nous pouvons la négliger. Ce qui implique que nous pouvons appliquer quelques rotations pour remettre ce vecteur unitaire dans le système d'axes W (centré sur B) à partir de C pour obtenir $\mathbf{u}_W$. En supposant que le point (u, v) provient de l'image rectifiée il suffit de suivre l'équation suivante :

$$\mathbf{u}^W = \mathbf{R}_B^W \mathbf{R}_C^B \mathbf{u}^C = \mathbf{R}_B^W \mathbf{R}_C^B \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4.7}$$

où $\mathbf{K}$ est la matrice des paramètres intrinsèques (eq. 2.3). Pour se rendre au rotor, il suffit donc de commander une vitesse dans la direction $\mathbf{u}^W$. Puisque nous n'avons pas de mesure de distance entre l'éolienne et l'UAV, la vitesse demeure constante ; elle peut être régulée que lorsque la surface entre dans la portée du capteur de proximité.

4.2.2 Approche en remontant la tour

Dans cette méthode le véhicule est placé à proximité de la tour de telle sorte que le scanner laser puisse capter la forme de la tour et de la discontinuité introduite à l'endroit où la tour rencontre le rotor. Cette méthode est plus simple et plus fiable que l'approche visuelle, mais elle repose sur deux hypothèses :

1. Le véhicule est bien placé par l'opérateur devant l'éolienne de telle sorte que les capteurs soient perpendiculaires au plan sur lequel les pales reposent.
2. Aucune pale n'est placée devant la tour, et l'angle de chaque pale est connu d'avance.

Le principe opérationnel est simple et illustré dans la Figure 4.6. Le scanner laser horizontal est utilisé pour centrer l'UAV devant la tour et garder une distance de sécurité alors que le scanner vertical est utilisé pour détecter le rotor. Pour ce faire, il faut tout d'abord projeter les mesures de distances à un nuage de points 3D au moyen de l'équation 4.8 où r_i est une distance, θ_i l'angle correspondant à la mesure i, et n le nombre de mesures retournées par le laser. Nous notons que cette méthode requiert peu de mesures laser et donc il n'y a pas besoin d'un lidar sophistiqué. Par exemple le lidar que nous utilisons ne fournit que huit

mesures.

$$\mathbf{p}_{Li} = \begin{bmatrix} p_{i_x}^L \\ p_{i_y}^L \\ p_{i_z}^L \end{bmatrix} = \begin{bmatrix} r_i \cos\theta_i \\ r_i \sin\theta_i \\ 0 \end{bmatrix} \text{ où } i = 1, \ldots, n \tag{4.8}$$

L'équation 4.8 nous donne des points dans le repère du laser correspondant, puisque les

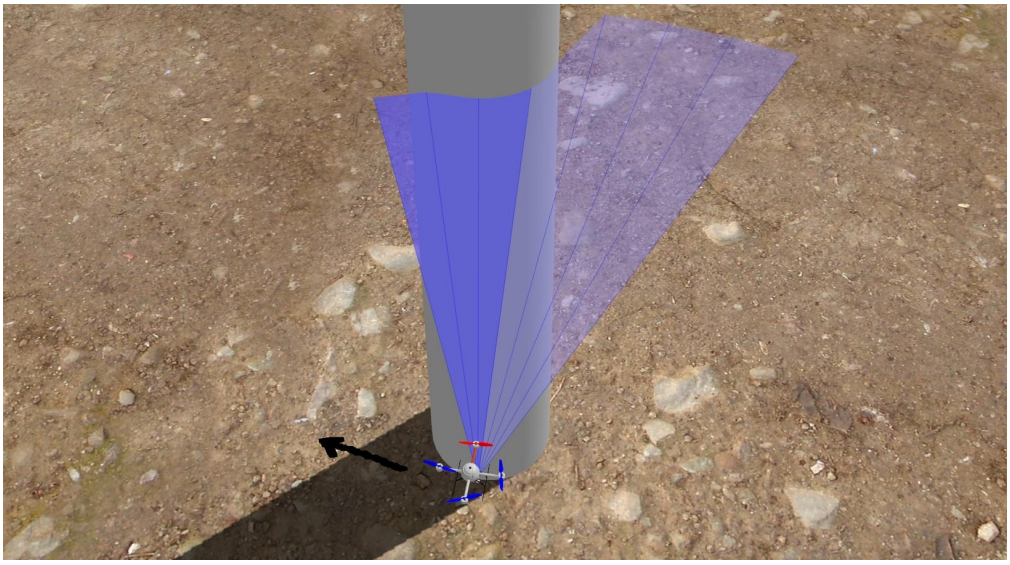


Figure 4.6 Principe d'opération de l'approche du rotor par laser.

matrices de transformées homogènes $\mathbf{T}_{LH}^B$, $\mathbf{T}_{LV}^B$ ainsi que la rotation $\mathbf{R}_B^{BT}$ sont connues, il suffit de transformer chaque $\mathbf{p}^L$ au repère BT suivant l'équation 4.9.

$$\begin{bmatrix} \mathbf{p}_i^{BT} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_B^{BT} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{T}_L^B \begin{bmatrix} \mathbf{p}_i^L \\ 1 \end{bmatrix} \tag{4.9}$$

Une fois ces mesures transformées dans le bon repère, nous pouvons appliquer une loi de contrôle de type PID sur les axes x et y en plus d'une vitesse constante v_z en z pour calculer la commande de vitesse $\mathbf{v}(t)$ dans le repère BT à envoyer à l'UAV tel que présenté dans l'équation 4.10.

$$\mathbf{v}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix} = \begin{bmatrix} K_{p_x} e_x(t) \\ K_{p_y} e_y(t) \\ v_z \end{bmatrix} + \begin{bmatrix} K_{ix} \int e_x dt \\ K_{iy} \int e_y dt \\ 0 \end{bmatrix} + \begin{bmatrix} K_{dx} \dot{e}_x(t) \\ K_{dy} \dot{e}_y(t) \\ 0 \end{bmatrix} \tag{4.10}$$

$$e_x = \frac{1}{n}\sum_{i=1}^{n} p_{i_x}^{BT} \tag{4.11}$$

$$e_y = d - \min(p_{i_y}^{BT}) \tag{4.12}$$

L'équation 4.11 présente le terme d'erreur e_x calculé en moyennant la composante x des points $p_{BT\,i}$ provenant du laser horizontal pour déterminer si l'UAV doit bouger vers sa gauche ou sa droite pour se centrer devant la tour. L'équation 4.12 présente le terme d'erreur e_y qui prend la plus petite mesure en y pour connaître la distance par rapport à la surface de la tour et d est la distance de sécurité à garder par rapport à la tour.

La détection du rotor se fait au moyen d'une détection de discontinuité en faisant une convolution (voir l'équation 4.14) de la composante y des points du laser vertical avec un filtre à dérivée de gaussienne $\Gamma(x)$ discrétisé sur n points dans le domaine $x \in [-5\sigma, 5\sigma]$ généré par l'équation 4.13. Une discontinuité est détectée si la réponse r du filtre est plus haut qu'un certain seuil à déterminer de façon empirique.

$$\mathbf{\Gamma}(x) = -\frac{x}{\sigma^3\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \tag{4.13}$$

$$r = p_y^{BT} * \mathbf{\Gamma}(x), \tag{4.14}$$

Pour contrer le bruit de mesure et s'assurer que la détection est valide, le véhicule ne s'arrête que si la détection s'active plusieurs fois dans un certain laps de temps. Le masque de convolution est de taille 1×8 et $\sigma = 0.5$.

4.3 Suivi des pales

Une fois devant le rotor, le véhicule utilise l'information sur l'angle des pales acquise au préalable (visuellement ou par entrée utilisateur) pour savoir dans quelle direction se diriger. Il est important à cette étape de garder une grande distance de sécurité ($d_{grand} = 8$ m ou plus) du rotor puisque celui-ci, constitué de multiples lignes électriques haute tension et d'une grande quantité de métal, peut causer de l'interférence magnétique avec le magnétomètre de l'UAV qui peut potentiellement résulter en une perte de l'orientation. Cette distance peut être diminuée à $d_{petit} = 4$ m lors du suivi une fois que le véhicule commence à s'éloigner du rotor.

Dans l'étape de suivi des pales, le principe est similaire à la montée de la tour. Le robot cherche à suivre la pale à une vitesse constante tout en restant à une distance désirée de la pale. Nous présentons deux approches possibles, la première dans la Section 4.3.1 par nuage de points applicable à des scanners lasers 3D ou un capteur par vision active, la deuxième dans la Section 4.3.2 n'usant que des scanners lasers 2D. Suivant la machine à états de la mission, une fois la fin de la pale détectée, le robot revient aux coordonnées GPS où le rotor a été détecté initialement. Puisque les pales d'une éolienne sont droites ou courbées vers l'avant, une trajectoire en ligne droite entre la fin de la pale et le devant du rotor est garantie d'être libre d'obstacles.

4.3.1 Suivi par nuage de points

Dans cette méthode, nous reprenons l'algorithme proposé dans le Chapitre 3 pour le suivi de mur par UGV. L'algorithme est d'ailleurs plus simple puisqu'il n'y a aucun sol à filtrer du nuage de points. Cependant l'avantage majeur de cette méthode est qu'avec le calcul de la normale de la surface il devient possible de toujours placer le véhicule perpendiculaire à celle-ci, améliorant ainsi la qualité des images prises pendant l'inspection.

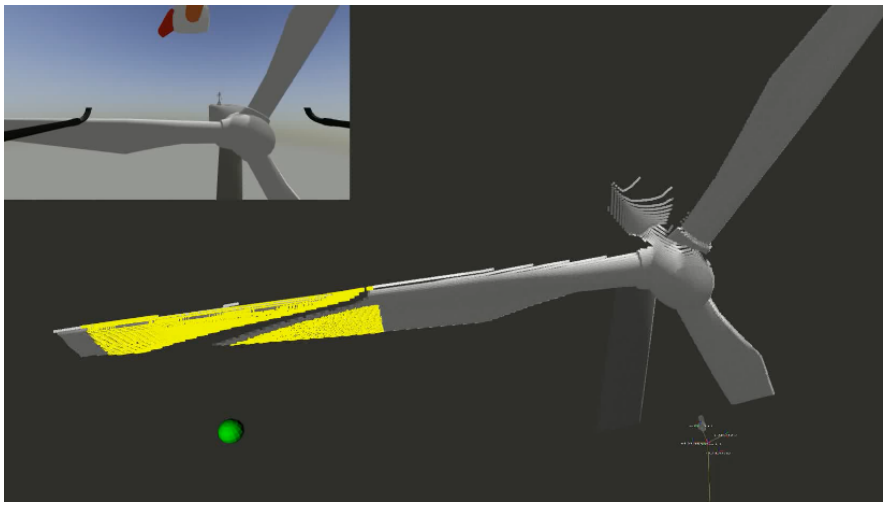


Figure 4.7 Exemple de suivi de pale par nuage de points.

Le bout de la pale est détectée de la même façon qu'un mur à angle aigu est détecté dans la section 3.2. À mesure que le véhicule avance vers le bout de la pale, le nuage de points $\mathcal{S}$ devient de plus en plus petit, ainsi il devient possible de détecter le bout une fois que $\mathcal{S}$ passe un certain seuil. Dans la Figure 4.7, les points blancs sont le nuage de points perçu par l'UAV alors que les points jaunes représentent la tranche sélectionnée $\mathcal{S}$. La boule verte est la commande de position envoyée à l'UAV, perpendiculaire à $\mathcal{S}$.

4.3.2 Suivi par scanner laser 2D

Le suivi par scanner laser se fait de la même façon que la montée de la tour au moyen de deux scanners lasers perpendiculaires l'un à l'autre. Suivant les mêmes formules de projection 2D à 3D présentées dans la section 4.2.2, le véhicule tente de se centrer par rapport à la pale en moyennant les points du nuage résultant tout en appliquant une vitesse constante V_C dans la direction estimée de suivie. Dépendamment de l'angle de la pale seulement l'un des deux lasers est utilisé pour le suivi.

La loi de commande est donc encore un PID où les termes d'erreurs sont calculés à partir de la projection laser et la sortie est une commande de vitesse dans le repère BT.

$$\mathbf{u}(t) = \mathbf{V}_C(t) + \mathbf{K}_P\mathbf{e}(t) + \mathbf{K}_I \int \mathbf{e}(t)dt + \mathbf{K}_D\dot{e}(t) \tag{4.15}$$

L'erreur $\mathbf{e}(t)$ est calculée différemment selon la direction de la pale. En pratique, on peut utiliser la moyenne du laser horizontal si la pale est à $\pm 15°$ de la verticale et le laser vertical dans tous les autres cas. Pour une pale horizontale nous aurions par exemple

$$\mathbf{e}(t) = \begin{bmatrix} e_x(t) \\ e_y(t) \\ e_z(t) \end{bmatrix} = \begin{bmatrix} 0 \\ d - \min(p_{i_y}^{BT}) \\ \frac{1}{n}\sum_{i=1}^{n} p_{i_z}^{BT} \end{bmatrix} \text{ avec } i = 1, \ldots, n \tag{4.16}$$

Dans le cas où une erreur de capteur survient, par exemple si la pale devient assez mince pour passer entre deux rayons du laser, on répète la commande de vitesse précédente $\mathbf{u}(t) = \mathbf{u}(t-1)$. Ceci implique qu'une fois que le véhicule passe le bout de la pale, il continuera à voler dans le vide. La détection de bout de pale se fait donc aisément en comptant le nombre de lectures vides dans un certain laps de temps avec un seuil à ajuster en prenant en compte que des lectures vides peuvent survenir pendant le suivi de la pale.

4.4 Implémentation

Pour l'implémentation, la majorité de l'effort de développement a été dédiée à l'approche purement par scanner laser suivant les sections 4.2.2 et 4.3.2 puisqu'elles s'avèrent être les plus fiables, dépendent de moins de paramètres à ajuster manuellement, et sont peut coûteuses en termes de matériel comparé à une approche par scanner laser 3D. Le suivi par nuage de point de la section 4.3.1 a aussi été implémenté séparément pour nous permettre de vérifier la possibilité d'utiliser un capteur de profondeur. Cependant, nous posons l'hypothèse que cette méthode ne fonctionnerait pas avec une caméra stéréo dû à la surface sans texture de l'éolienne ni avec un capteur fonctionnant par *Time of Flight* de par leur portée limitée. De plus, nous implémentons aussi la méthode de Stokkeland modifiée décrite à la section 4.2.1 pour la détection du rotor, mais à la vue des mauvais résultats obtenus nous n'avons pas poursuivi l'implémentation du contrôleur correspondant.

La simulation est donc développée en prévision d'un quadricoptère md4-1000 de Microdrones équipé de deux lidars LeddarVu8 de LeddarTech, une caméra stéréo Zed de ZedLabs et un ordinateur de bord Nvidia TX2. Les lidars Vu8 ont été sélectionnés avec un angle de vue de 48° et ont la propriété particulière d'émettre des segments au lieu de faisceaux colmatés à la façon des lidars traditionnels. Par conséquent, il devient improbable en pratique qu'une pale d'éolienne puisse se perdre entre deux rayons du lidar.

L'intégralité du logiciel a été développé en C++ à l'aide de l'interlogiciel Robot Operating System (ROS) incluant les pilotes pour les divers capteurs et la communication par UART au md4-1000. Le traitement d'images repose sur la librairie OpenCV (Itseez, 2017) et la machine à états faisant la gestion de la mission, présentée dans l'annexe B, a été écrite à l'aide de la librairie Boost *Meta State Machine* (Schling, 2011). En tout temps, la machine, que l'on nomme `Commander` garde un pointeur vers un objet implémentant l'interface `ControllerInterface`. Ces contrôleurs sont instanciés et détruits dynamiquement à chaque changement d'état, nous permettant ainsi d'avoir accès à une grande quantité de contrôleurs spécifiques tout en gardant l'utilisation de la mémoire vive à un minimum. Le `Commander` calcule les commandes de vitesse à une fréquence de 50Hz, la même que la fréquence maximale de l'odométrie. Quand une mise à jour d'odométrie ou des lidars est reçue, elle est mise dans un tampon en attente du prochain cycle de calcul.

La Figure 4.8 présente les contrôleurs utilisés dans le logiciel. Nous avons en particulier le `PositionController` pour la phase de retour au rotor, le `LidarBladeFollower` pour le suivi de pale et le `TowerAscentController` pour la montée de la tour. Le `DirectionController` n'est utilisé que pendant de courts instants de contrôle en boucle ouverte, par exemple la

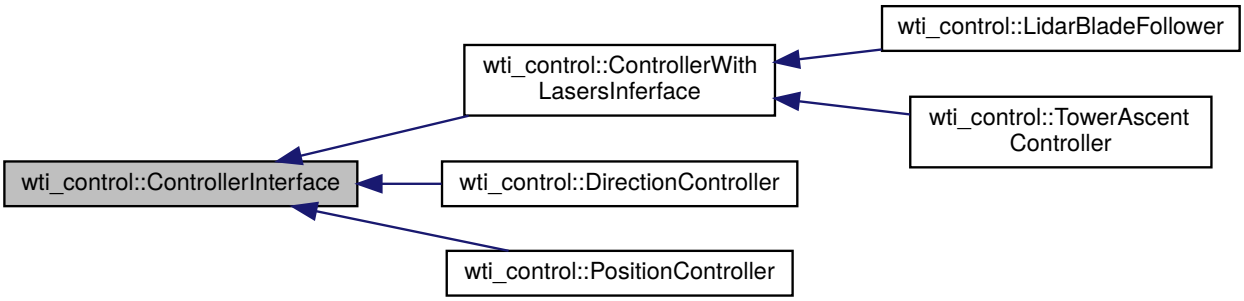


Figure 4.8 Diagramme de classe des différents contrôleurs utilisés par la machine à états.

manœuvre pour éviter la zone d'interférence magnétique devant la nacelle.

4.4.1 Résultats en simulation

La simulation se base sur celle pour véhicules multi rotors nommée RotorS développée par Furrer *et al.* (2016) dans l'environnement Gazebo [1]. Puisque le micrologiciel effectuant le contrôle de vol dans les véhicules de Microdrones demeure un secret industriel la simulation a été développée au moyen de véhicules déjà présents dans RotorS. Le but n'est donc pas de réaliser une simulation précise de la physique du véhicule. On se concentre plutôt sur la simulation des capteurs et des comportements autonomes de haut niveau.

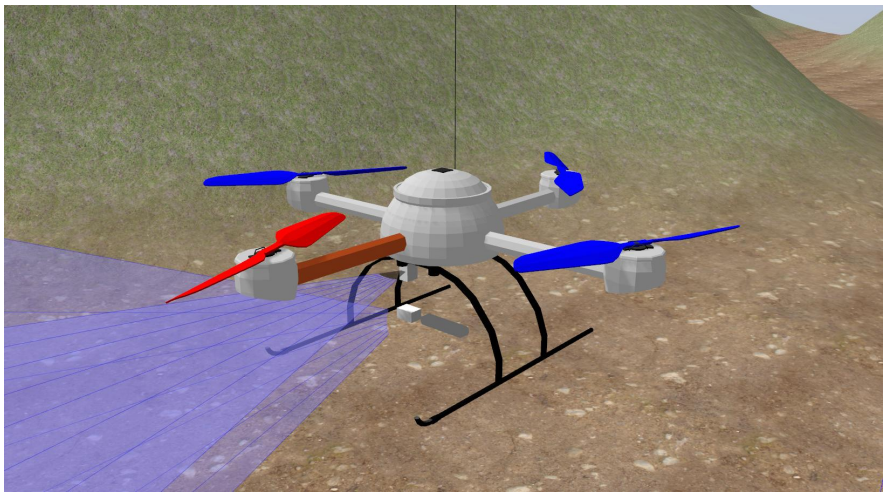
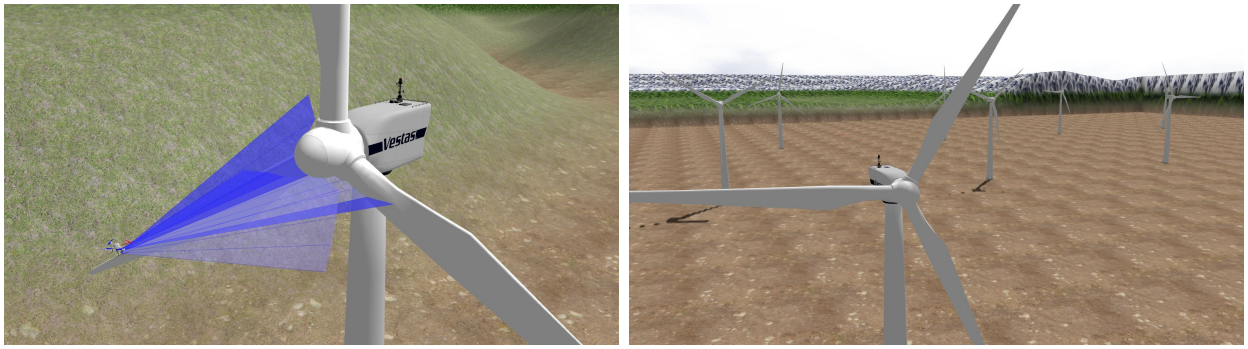


Figure 4.9 Véhicule md4-1000 en simulation. Les rayons bleus sont les rayons projetés par les scanner lasers LeddarVu8.

Plusieurs environnements ont été implémentés incluant un monde simulant un parc éolien

1. http ://gazebosim.org

avec plusieurs éoliennes alignées (Figure 4.10 (B)) et un monde avec une seule éolienne avec un montagne au loin (Figure 4.10 (A)). La simulation suit le pire cas possible d'un parc éolien respectant les consignes du Département de l'Environnement du Royaume-Uni demandant un minimum de trois diamètres de rotor d'espacement entre chaque éolienne (Department of Environment, 2009). Il faut noter que suite à des discussions avec des travailleurs de l'industrie, l'un des défauts de la simulation est que les pales des éoliennes sont normalement inclinées vers l'arrière de telle sorte qu'elles sont parallèles au sens du vent. Ceci permet au rotor de rester stationnaire lors des inspections. Notre algorithme reste toujours valide mais il ne permet que d'inspecter le bord d'attaque des pales.



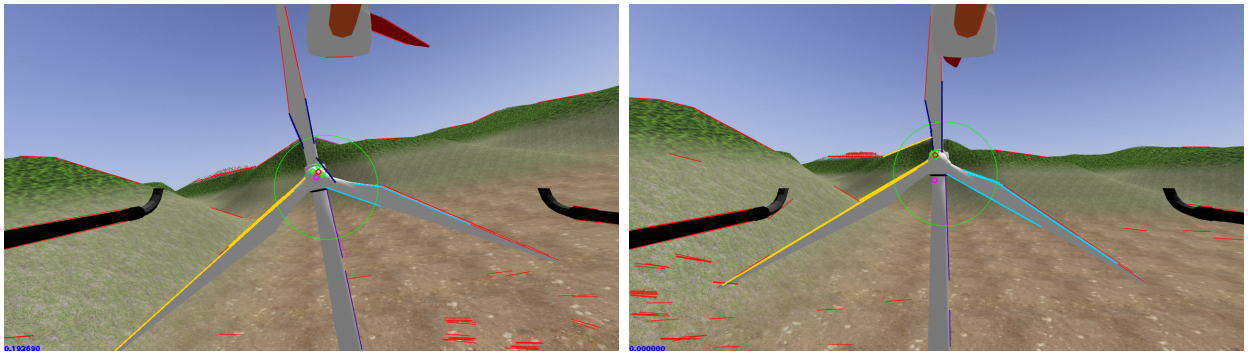
(A) (B)

Figure 4.10 Environnements de simulation.

Résultats par traitement d'images

Tout d'abord, nous testons en simulation la méthode de Stokkeland modifiée. Nous pouvons observer dans la Figure 4.11 (A) une détection réussie en présence d'un angle de roulis et dans la Figure 4.11 (B) une détection lorsque l'horizon est présent près du rotor. Le grand cercle vert représente le rayon de recherche dans lequel les segments de pale sont recherchés, les petits cercles verts sont les intersections des projections des pales, le cercle rouge est la moyenne des intersections (le centre du rotor) et le cercle rose est la sortie du filtre de Kalman.

Suite à plusieurs expérimentations, il devient difficile d'apprécier cette méthode et nous démontrons qu'une approche entièrement visuelle sans information au préalable n'est pas pratique. Dans nos tests, en simulant une trajectoire d'avance manuellement à l'aide d'une manette, le taux de détection varie grandement entre 10% et 60%, et dépend fortement du réglage des divers paramètres des filtres gaussiens, de la détection des contours et des lignes et des divers paramètres de distance de recherche. De plus, une fois les paramètres réglés



(A) (B)

Figure 4.11 Détections réussies en simulation.

pour augmenter le taux de détection, ceux-ci deviennent invalides lorsque l'on modifie les paramètres de l'environnement de simulation tels que l'angle des pales, la texture du sol, les nuages dans le ciel et la position de la source de lumière. Un autre problème rencontré est la difficulté de choisir les paramètres du flou gaussien tel que la texture de l'arrière-plan soit atténuée tout en préservant les contours de l'éolienne pour l'algorithme de Canny. En somme, il est impossible de régler les paramètres au préalable pour accommoder toutes les conditions possibles qui seront rencontrées sur le terrain.

Nous profitons de la présence de la caméra stéréo ZED simulée pour tester le fonctionnement de la résolution de la profondeur sur la surface de l'éolienne. Nous pouvons observer dans la Figure 4.12 que la surface blanche de l'éolienne ne comporte pas assez de texture pour bien calculer la résolution stéréo. Résultat inattendu, le calcul de profondeur fonctionne parfois près de la bordure de l'objet où la transition entre la pale blanche et l'arrière-plan donne une caractéristique pouvant être mise en correspondance dans l'image de droite. Par contre, lorsque la pale est horizontale, elle devient quasi-invisible du fait que la bordure est parallèle à la géométrie épipolaire. Ce problème est bien connu par les chercheurs utilisant la vision stéréoscopique pour le calcul de profondeur. Certains tels que Meier *et al.* (2017) ont récemment proposé une solution possible à l'aide de caméras stéréo orthogonales ; cependant la résolution de ce problème sort du cadre de ce projet.

Résultats du suivi par laser

La mission d'inspection entièrement par laser se déroule avec succès dans une grande variété de scénarios. Dans cette section, nous présentons et analysons l'un des vols effectués sur une éolienne avec pour seule information *a priori* que l'une des pales a été placée vers le haut. Dans la Figure 4.15, on peut observer la trajectoire entière exécutée par l'UAV avec

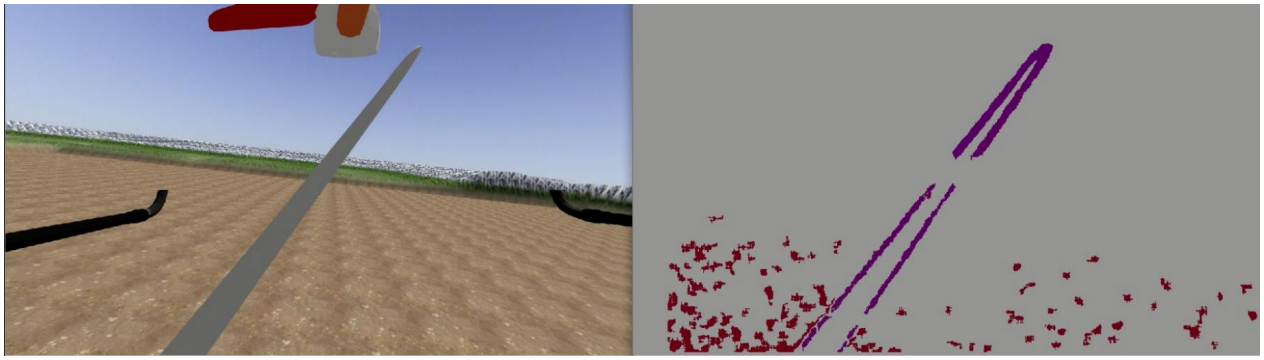


Figure 4.12 Échec de la résolution de profondeur par correspondance de blocs.

les différentes phases séparées par couleur. Outre une petite discontinuité dans la trajectoire de la troisième pale, la majorité du vol s'exécute de façon continue.

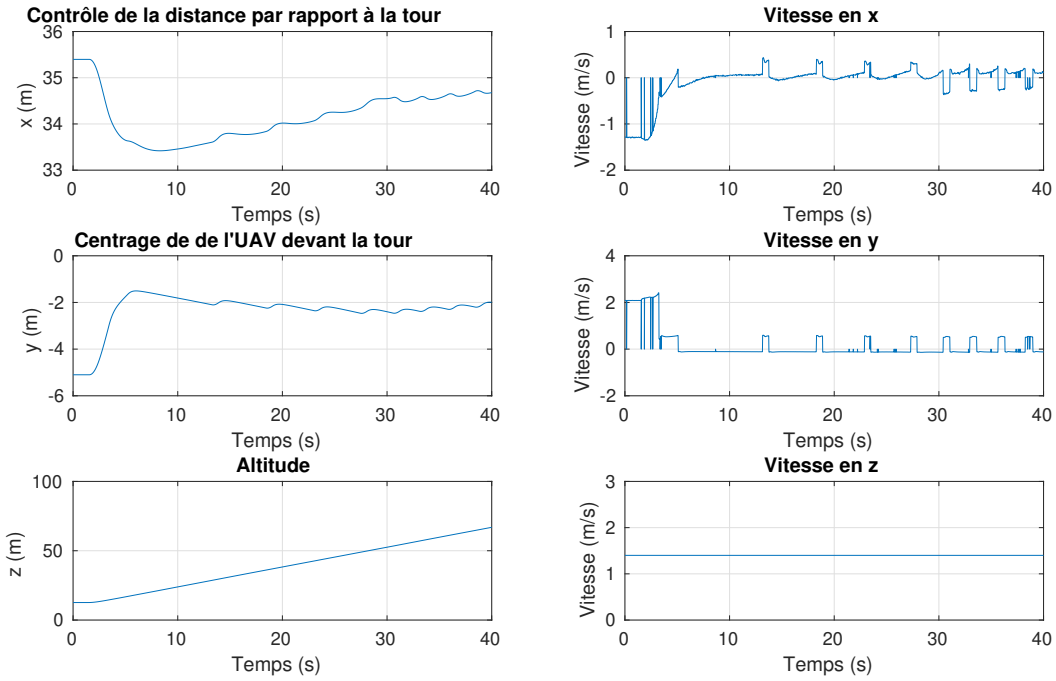


Figure 4.13 Trajectoire suivie pour la montée de la tour et les commandes de vitesse associées.

Dans la Figure 4.13, nous pouvons voir la décomposition de la trajectoire suivie pour monter la tour. L'UAV commence légèrement décentré par rapport à la tour et tente immédiatement de se centrer. La vitesse commandée en z est constante et on voit que l'altitude augmente de

façon constante. Par contre, on peut voir que les déplacements en x et en y sont légèrement saccadés. En x, la raison est une combinaison de gains PID légèrement agressifs et du fait que le diamètre de la tour change au fur et à mesure qu'elle monte. En y, le problème provient plutôt de la faible résolution du scanner laser. Avec seulement 8 points, la bordure de la tour se retrouve entre deux rayons du laser. Les différentes bosses dans la courbe v_y correspondent aux instants où l'un des rayons adjacents au centre frappe la tour. À cet instant le terme d'erreur devient non nul et le robot bouge latéralement.

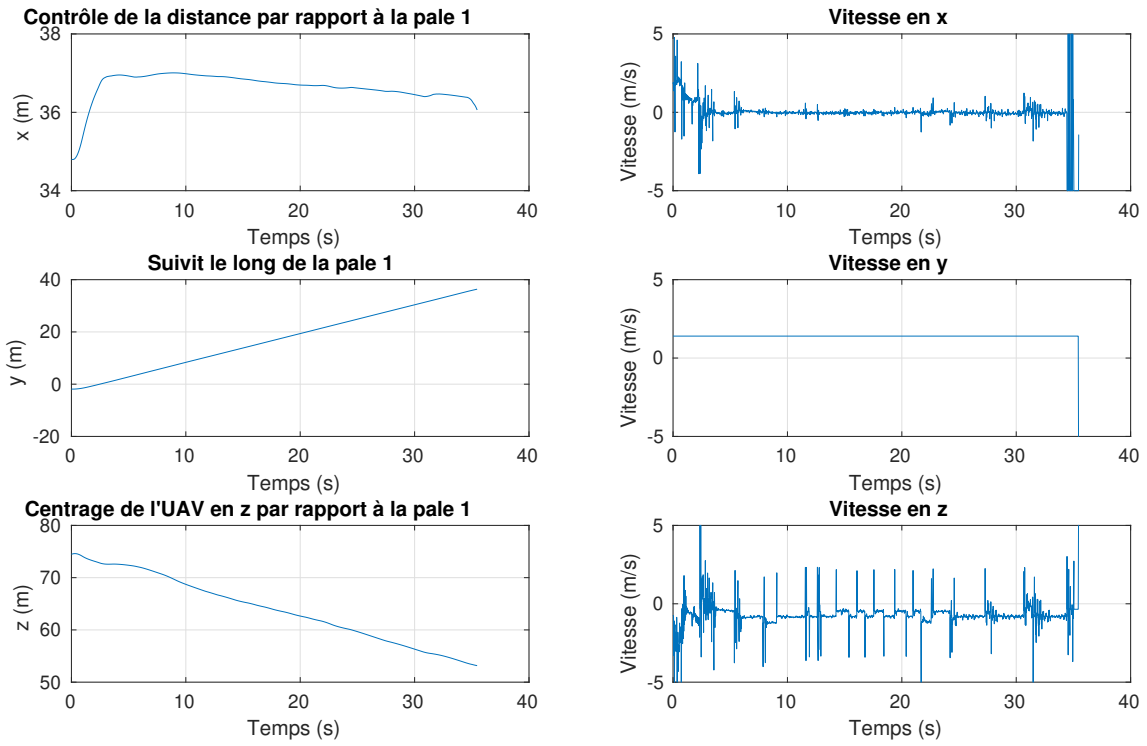


Figure 4.14 Trajectoire pour le suivi de la pale 1.

La Figure 4.14 présente la trajectoire et les commandes de vitesses pour le suivi de la pale 1. Nous voyons que les vitesses en z semblent saccadées, mais ceci est encore dû au petit nombre de rayons du scanner laser. Chaque petit pic est en fait un instant où un rayon supplémentaire du laser parvient à toucher la surface de la pale et influencer le calcul de son centre. Vers la fin de la courbe, v_x nous pouvons observer une oscillation entre les commandes 0, $+5$ et -5 mètres par seconde. Ceci est dû au fait que le bout de la pale est assez petit pour passer entre deux rayons du scanner laser, combiné au mouvement de tangage du véhicule pour avancer et reculer, le laser parvient momentanément à toucher la pale. En réalité ceci ne devrait pas

être un problème, car le LeddarVu8 ne souffre pas des angles morts entre ses rayons. Malgré cela, le robot poursuit son chemin jusqu'à ce que le bout de la pale soit officiellement détectée par le compteur de lectures vides.

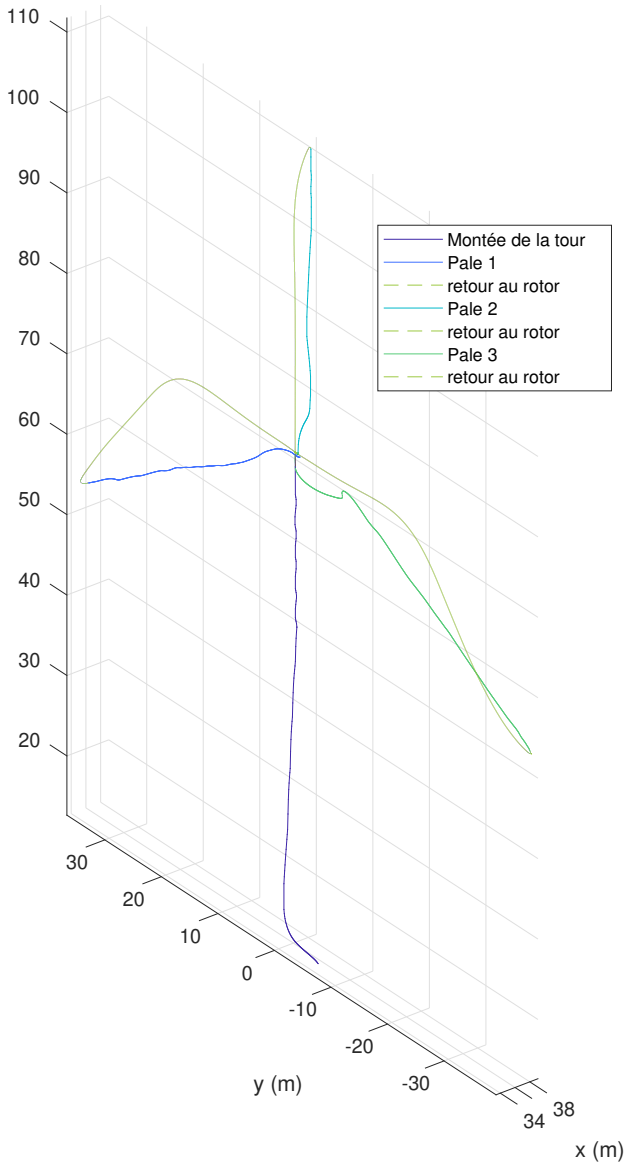


Figure 4.15 Trajectoire complète d'inspection.

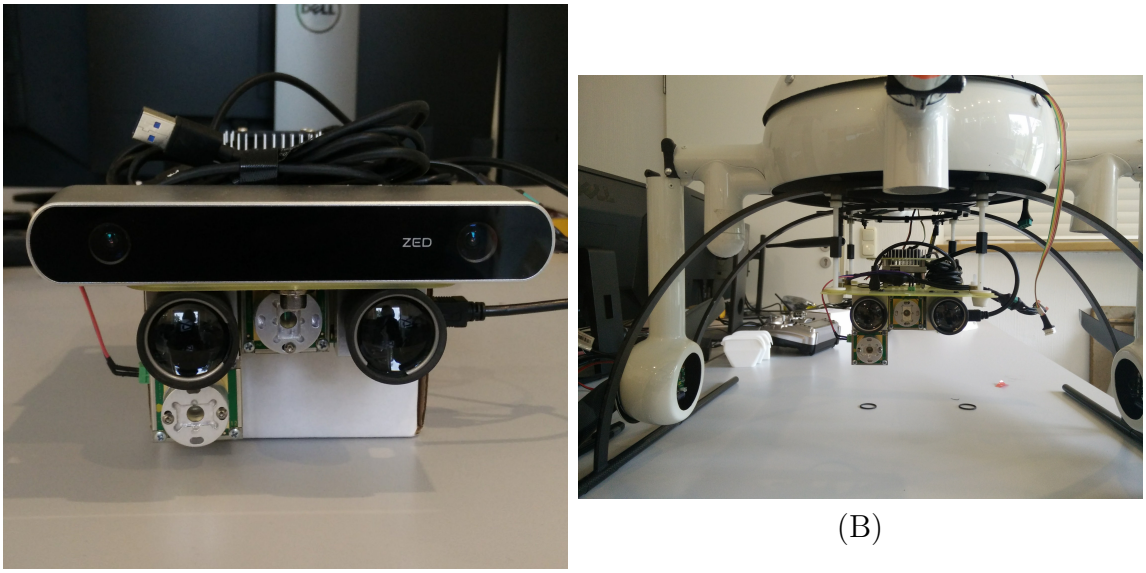
4.4.2 Résultats de tests sur le terrain

Figure 4.16 Le md4-1000 en vol inspectant l'une des pales.

Suivant les spécifications initiales, l'intégration des capteurs s'est faite sur un md4-1000 pris en photo dans la Figure 4.17 pour faire un test de vol manuel nous permettant de confirmer certaines de nos hypothèses au niveau de la vision stéréo. Le système d'exploitation Ubuntu 16.04 a été installé sur l'ordinateur Nvidia TX2 avec des modifications apportées au noyau Linux fourni par Nvidia afin de permettre au TX2 de s'interfacer avec les différents périphériques montés sur la carte mère AuvIdea J140. Les LeddarVu8 et la caméra ZED sont branchés par USB3 alors que la communication au md4-1000 se fait par UART.

Les tests de vol ont été effectués au parc éolien Pierre-de-Saurel où la hauteur de la tour est de 100 mètres et le diamètre du rotor incluant les pales est de 92.5 mètres. La Figure 4.16 nous donne un ordre de grandeur de la structure par rapport à l'UAV.

La caméra stéréo s'avère à avoir une performance extrêmement variable à travers le vol tel qu'on peut le voir dans la Figure 4.18. Il est difficile de corréler exactement les caractéristiques de la scène au succès ou à l'échec de la résolution de profondeur. Dans certains cas, une pale diagonale est invisible alors que dans le dernier exemple, elle est parfaitement visible. On voit aussi que le ciel clair est parfois résolu par erreur. Par conséquent, nous prouvons notre hypothèse initiale et en concluons qu'une caméra stéréo est inutilisable dans le contexte de



(A) (B)

Figure 4.17 Véhicule md4-1000 avec les capteurs et l'ordinateur de bord intégrés. (A) Caméra ZED et les télémètres lasers LeddarVu8. (B) La charge utile installée sur le robot (sans la ZED), avec la Nvidia TX2 visible.

l'inspection d'éoliennes.

En ce qui concerne l'approche par laser il n'a pas été possible de tester le système sur le terrain en raison de différentes circonstances et d'une erreur logicielle survenue le jour du test. Plus de temps serait requis pour faire l'intégration et le réglage des gains PID sur le vrai véhicule. Par contre, puisque l'entièreté du logiciel est déjà écrite en C++ et prête à installer sur le véhicule, il ne suffirait que de changer quelques branchements au niveau des flux de données pour faire les tests sur le terrain.

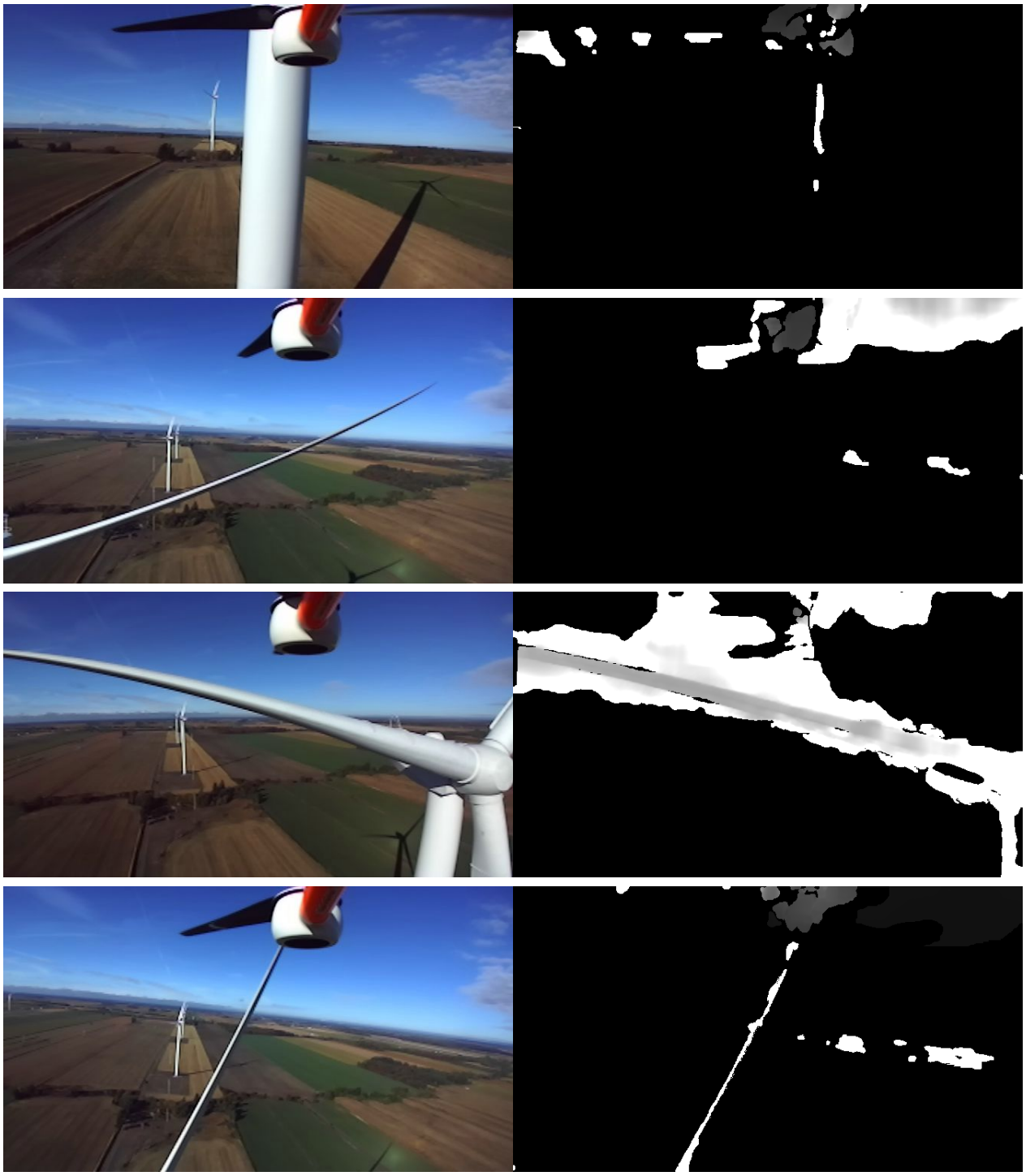


Figure 4.18 Exemples de succès et d'échecs de perception de distance. À gauche, les images RGB de la caméra de gauche, et à droite, les images de profondeur ramenées sur la caméra de gauche.

4.5 Discussion des résultats et travaux futurs

Bien que nous n'avons pas pu valider notre approche sur le terrain, les résultats en simulation demeurent prometteurs et l'inspection d'éoliennes reste un projet important à poursuivre dans le futur. Par contre, suite à nos tests de vol manuels sur le terrain, nous réalisons maintenant que la méthode d'inspection bénéficierait de plusieurs changements.

Tout d'abord, pour qu'un UAV entre 1 et 25 kg utilisé à des fins non récréatives soit exempt du besoin d'un Certificat d'Opération Aériennes Spécialisées (COAS) par Transport Canada, il doit demeurer en dessous d'une altitude de 300 pieds (91 mètres) (Transports Canada, 2016). Puisque les grandes éoliennes ont leur nacelle à une altitude de 100 mètres ou plus, ceci impliquerait que chaque fois qu'un inspecteur voudrait aller travailler, il faudrait appliquer pour un COAS à l'avance, ce qui peut prendre beaucoup de temps à être approuvé. Sans quoi, l'UAV n'aurait pas le droit d'inspecter les pales autres que celles pointant vers le bas.

Il reste néanmoins une alternative possible qui comporterait plusieurs avantages sur le système proposé à condition de remettre en question les modalités de la participation de l'inspecteur et du budget alloué. Le système pourrait faire une inspection en 3 dimensions d'une seule pale à la fois pointée vers le bas suivant une trajectoire présentée dans la Figure 4.19. En commençant sur le côté de la pale l'UAV longerait la pale vers bas. Il pivoterait ensuite pour remonter le bord d'attaque avant de pivoter une dernière fois pour longer le côté droit. Ceci demanderait des capteurs différents et nécessiterait que l'inspecteur fasse tourner les pales entre chaque vol. On aurait aussi l'avantage d'avoir de bien meilleures images puisque la caméra serait pointée perpendiculairement à la surface à inspecter.

Pour en revenir à la méthode que nous proposons, l'UAV bénéficierait grandement de l'utilisation de D-GPS ou de GPS-RTK à deux antennes. Le premier avantage serait qu'avec un positionnement plus précis, il serait possible de se rapprocher beaucoup plus de la structure et obtenir de meilleures images. Les distances utilisées en simulation prennent en compte que la précision du positionnement du md4-100 est typiquement entre $\sigma = 1.0$ et $\sigma = 1.5$ mètres. De plus, avec les systèmes à deux récepteurs, il n'y aurait plus d'inquiétudes à propos de la zone d'interférence magnétique devant la nacelle.

Bien que ce soit rare, il arrive parfois que la mission échoue lorsque l'UAV perd de vue la pale qu'il est en train de suivre. La cause est habituellement une instabilité lorsque le dernier faisceau du lidar est près de la bordure de la pale. L'UAV tangue pour s'approcher ou reculer et le lidar perd ou regagne la pale. Ce mouvement peut se répéter jusqu'à ce que l'oscillation fasse perdre la pale complètement. Ce problème peut être palié en introduisant une meilleure gestion du cas où le lidar tombe en faute, une procédure de recherche lorsque la pale est

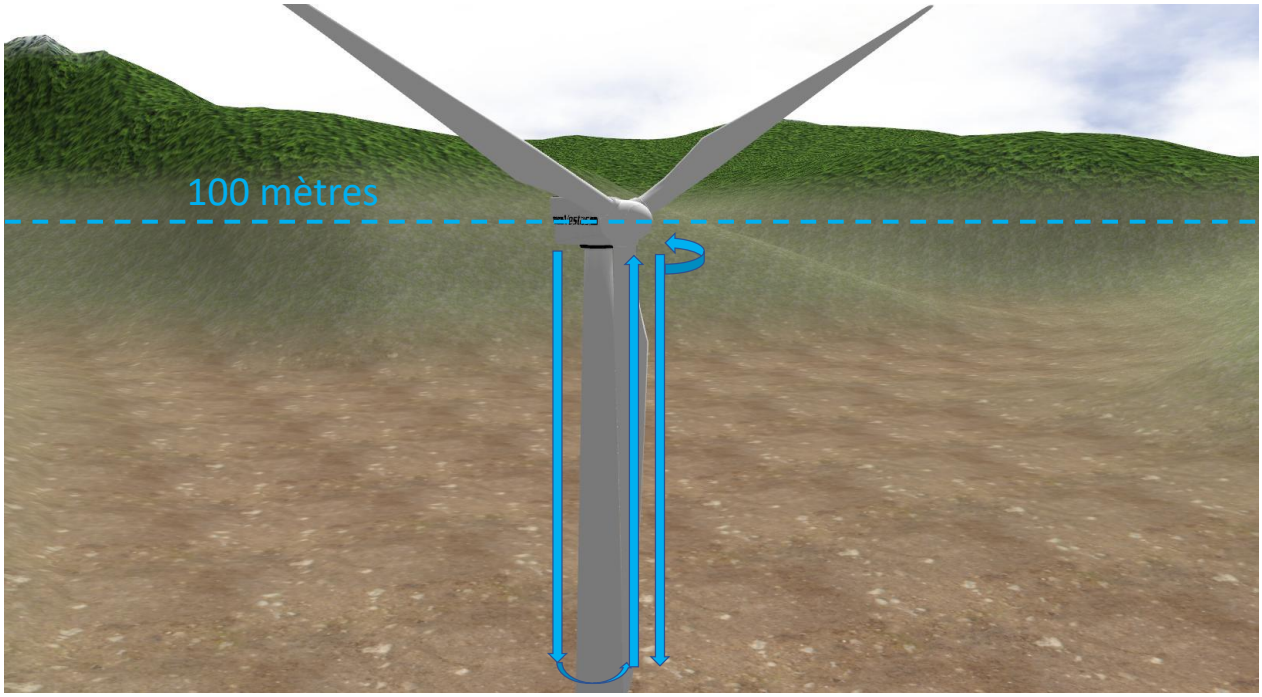


Figure 4.19 Trajectoire alternative pour l'inspection d'une pale à la fois sans besoin de COAS.

perdue et une limite sur l'accélération à la sortie des contrôleurs.

En ce qui concerne les résultats du traitement visuel, il est clair que la méthode proposée ne mérite pas plus d'investigations. Certaines possibilités pourraient être étudiées par exemple après une détection réussie, les descripteurs de caractéristiques tels que *Oriented FAST and Rotated BRIEF* (Rublee *et al.*, 2011) pourraient êtres extraits et suivi par un algorithme de flux optique (Lucas et Kanade, 1981). Ceci donnerait une deuxième mesure au filtre de Kalman potentiellement plus fiable. Une autre possibilité serait de remplacer la procédure de détection par un réseau de neurones entièrement convolutionnel qui calculerait conjointement la segmentation et l'orientation des pales. L'une des pistes prometteuses que nous avions commencé à étudier est l'utilisation de filtres non-linéaires tel qu'un filtre à particules pour le suivi et la localisation de l'éolienne (Sugandi *et al.*, 2009). En somme, nous n'écartons pas la possibilité de faire une inspection entièrement par vision, mais ceci devrait être fait au moyen de méthodes plus avancées que celles examinées dans le présent mémoire.

CHAPITRE 5 CONCLUSION

Dans ce mémoire, nous avons présenté deux méthodes, pour l'inspection de diverses infrastructures civiles, l'une par UGV et l'autre par UAV. À travers ceci, nous étudions aussi l'utilisation d'une variété de capteurs permettant la navigation d'un robot dans un environnement inconnu. Dans ce chapitre, nous faisons un survol des méthodes proposées et nous proposons des pistes pour la poursuite du travail présenté.

5.1 Synthèse des travaux

Le système d'inspection par UGV permet de cartographier la surface visible d'une structure tout en minimisant l'erreur de localisation sur le module de SLAM utilisé. L'UGV perçoit son environnement au moyen d'un bras articulé sur lequel nous installons une caméra de profondeur. La carte construite par la caméra permet ensuite d'appliquer la méthode des champs de potentiel pour déplacer le robot à l'endroit désiré. Notre stratégie consiste en une exploration de périmètre permettant d'effectuer une fermeture de boucle le plus tôt possible. Une fois la boucle fermée et la contrainte sur le graphe de poses imposé, l'exploration des cavités débute pour compléter le modèle en construction. Les cavités sont détectées par une analyse des voxels d'une OctoMap pour décerner les frontières entre l'espace connu et l'espace inconnu. Nous prouvons la validité de la méthode au travers de multiples expériences tant en simulation qu'en expérimentation réelle. Le travail a eu pour point culminant la publication d'un article dans le journal *IEEE Transactions on Automation Science and Engineering* (Ramanagopal *et al.*, 2017).

Nous proposons aussi un système d'inspection par UAV permettant de survoler la surface des pales d'une éolienne à l'arrêt. Au moyen de deux lidars 2D, une méthode de contrôle PID est appliquée pour centrer le véhicule sur la partie de l'éolienne sous inspection. Une simple machine à états dirige l'UAV pour décoller du sol et inspecter une grande partie de la surface avant de la structure. La méthode est validée à travers de multiples simulations dans l'environnement Gazebo. Cependant, plus de travail serait requis pour terminer l'implémentation sur un véhicule réel. Nous étudions aussi l'utilisation de vision par ordinateur pour la détection et l'approche de l'éolienne. Les résultats expérimentaux indiquent que la poursuite de cette méthode demanderait l'exploration d'algorithmes entièrement différents.

5.2 Limitations de la solution proposée et améliorations futures

Dans les sections 3.4.3 et 3.5, nous avons fait un survol des cas d'échec possible de la méthode d'inspection par UGV. En particulier, notre algorithme est sensible aux irrégularités présentes à la surface de la structure, telles que des fenêtres ou des trous dans les murs. En présence de cette sorte de défaut, le caméra de profondeur peut capter une partie de l'intérieur de la structure, ce qui fausse le calcul de la prochaine pose objectif lors du suivi de mur. Bien qu'un seuillage sur le nuage de points est possible pour mitiger ce problème, il peut arriver qu'une partie des points qui nous intéressent soient perdus dans le processus. Pour mitiger le problème il faudrait développer un moyen intelligent de segmenter le nuage de points capté pour n'inclure que la surface de la structure. Ce problème sera particulièrement important à résoudre pour déployer notre robot dans des missions d'inspection de chantiers où les surfaces de la structure seront incomplètes.

Dans la section 4.5, nous avons discuté des limites du système d'inspection par UAV. Le projet a été réalisé avec une simplification de la tâche où, seul le bord d'attaque des pales de l'éolienne devait être inspectées. Un système complet devrait être capable d'inspecter toutes les facettes d'une pale incluant les deux côtés, le bord d'attaque et le bord de fuite. De plus, la conception du véhicule multi-rotor devrait être revue pour éliminer le moteur dans le champ de vision de la caméra et pour permettre à la caméra de pointer vers le haut afin d'inspecter le dessous des pales.

Une axe de recherche intéressante pour la poursuite de ce travail serait de résoudre le problème de l'approche de loin sachant par exemple seulement les coordonnées GPS de l'éolienne. Ceci serait la prochaine étape avant d'attaquer le développement d'une flotte d'UAV assurant l'entretien d'un parc éolien entier en effectuant des inspections en parallèles. Une autre piste de recherche pour la continuation de ce projet serait l'inspection d'éoliennes en mouvement.

RÉFÉRENCES

Acar, Ercan U. and Choset, Howie (2002). Sensor-based Coverage of Unknown Environments : Incremental Construction of Morse Decompositions. *The International Journal of Robotics Research*, *21*(4), 345–366.

Acar, Ercan U and Choset, Howie and Rizzi, Alfred A and Atkar, Prasad N and Hull, Douglas (2002). Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, *21*(4), 331–344.

Cuneyt Akinlar and Cihan Topal (2011). Edlines : A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, *32*(13), 1633 – 1642.

Bircher, Andreas and Alexis, Kostas and Burri, Michael and Oettershagen, Philipp and Omari, Sammy and Mantel, Thomas and Siegwart, Roland (2015). Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6423–6430.

Bircher, Andreas and Kamel, Mina and Alexis, Kostas and Oleynikova, Helen and Siegwart, Roland (2016). Receding Horizon "Next-Best-View" Planner for 3D Exploration. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1462–1468.

Borenstein, J. and Everett, H. R. and Feng, L. and Wehe, D. (1997). Mobile robot positioning : Sensors and techniques. *Journal of Robotic Systems*, *14*(4), 231–249.

Brilakis, Ioannis and Fathi, Habib and Rashidi, Abbas (2011). Progressive 3D reconstruction of infrastructure with videogrammetry. *Automation in Construction*, *20*(7), 884–895.

J. Canny (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698.

L. Carlone and G. C. Calafiore and C. Tommolillo and F. Dellaert (2016). Planar pose graph optimization : Duality, optimal solutions, and verification. *IEEE Transactions on Robotics*, *32*(3), 545–565.

H. Carvalho and P. Del Moral and A. Monin and G. Salut (1997). Optimal nonlinear filtering in gps/ins integration. *IEEE Transactions on Aerospace and Electronic Systems*, *33*(3), 835–850.

Howie Choset and Kevin M. Lynch and Seth Hutchinson and George A. Kantor and Wolfram Burgard and Lydia E. Kavraki and Sebastian Thrun (2005). *Principles of Robot Motion : Theory, Algorithms, and Implementations*. MIT Press, Pittsburgh, PA.

Chui, Charles K. and Chen, Guanrong (2017). *Kalman Filter : An Elementary Approach*, Springer International Publishing, Cham. 19–31.

Cieslewski, Titus and Kaufmann, Elia and Scaramuzza, Davide (2017). Rapid Exploration with Multi-Rotors : A Frontier Selection Method for High Speed Flight. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver.

D. Claus and A. W. Fitzgibbon (2005). A rational function lens distortion model for general cameras. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. vol. 1, 213–219 vol. 1.

DeepMap (2017). Hd maps for autonomous vehicles.

Department of Environment (2009). Best practice guidance to planning policy statement 18 'renewable energy'. `https://www.planningni.gov.uk/index/policy/planning_statements_and_supplementary_planning_guidance/planning_policy_statement_18__renewable_energy__best_practice_guidance.pdf`. (Accessed on 10/23/2017).

Dornhege, Christian and Kleiner, Alexander (2011). A frontier-void-based approach for autonomous exploration in 3d. *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE, 351–356.

Duda, Richard O. and Hart, Peter E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, *15*(1), 11–15.

Eldada, Louay (2016). Solid state lidar for ubiquitous 3d sensing. `http://on-demand.gputechconf.com/gtc/2016/presentation/s6726-louay-eldada-quanergy-systems.pdf`. (Accessed on 11/16/2017).

B. Englot and F. Hover (2010). Inspection planning for sensor coverage of 3d marine structures. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4412–4417.

Martin Ester and Hans-Peter Kriegel and Jörg Sander and Xiaowei Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, 226–231.

F. Fraundorfer and L. Heng and D. Honegger and G. H. Lee and L. Meier and P. Tanskanen and M. Pollefeys (2012). Vision-based autonomous mapping and exploration using a quadrotor mav. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4557–4564.

D. Fridovich-Keil and E. Nelson and A. Zakhor (2017). Atommap : A probabilistic amorphous 3d map representation for robotics and surface reconstruction. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 3110–3117.

Fadri Furrer and Michael Burri and Markus Achtelik and Roland Siegwart (2016). Robot operating system (ros). *Studies Comp.Intelligence Volume Number :625*, *The Complete Reference (Volume 1)*(978-3-319-26052-5), Chapter 23. ISBN :978-3-319-26052-5.

E. Galceran and R. Campos and N. Palomeras and M. Carreras and P. Ridao (2014). Coverage path planning with realtime replanning for inspection of 3d underwater structures. *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 6586–6591.

Google (2017). Google earth vr.

D. De Gregorio and L. Di Stefano (2017). Skimap : An efficient mapping framework for robot navigation. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2569–2576.

G. Grisetti and R. Kummerle and C. Stachniss and W. Burgard (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), 31–43.

G. Grisetti and C. Stachniss and W. Burgard (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, *23*(1), 34–46.

Grompone von Gioi, Rafael and Jakubowicz, Jérémie and Morel, Jean-Michel and Randall, Gregory (2012). LSD : a Line Segment Detector. *Image Processing On Line*, *2*, 35–55.

Heggem, Hans Erik (2017). *Autonomous Wind Blade Inspection*. Master thesis, Norges teknisk-naturvitenskapelige universitet.

Henry, Peter and Krainin, Michael and Herbst, Evan and Ren, Xiaofeng and Fox, Dieter (2014). Rgb-d mapping : Using depth cameras for dense 3d modeling of indoor environments. *Experimental robotics*. Springer, 477–491.

Hepp, Benjamin and Nießner, Matthias and Hilliges, Otmar (2017). Plan3D : Viewpoint and Trajectory Optimization for Aerial Multi-View Stereo Reconstruction.

Hernandez, Carlos and Bharatheesha, Mukunda and Ko, Wilson and Gaiser, Hans and Tan, Jethro and van Deurzen, Kanter and de Vries, Maarten and Van Mil, Bas and van Egmond, Jeff and Burger, Ruben and Morariu, Mihai and Ju, Jihong and Gerrmann, Xander and Ensing, Ronald and Van Frankenhuyzen, Jan and Wisse, Martijn (2017). *Team Delft's Robot Winner of the Amazon Picking Challenge 2016*, Springer International Publishing. 613–624.

Herout, Adam and Dubská, Markéta and Havel, Jirí (2013). *Variants of the Hough Transform for Straight Line Detection*, Springer London, London. 17–34.

Hert, Susan and Tiwari, Sanjay and Lumelsky, Vladimir (1996). A terrain-covering algorithm for an AUV. *Autonomous Robots*, *3*(2-3), 91–119.

W. Hess and D. Kohler and H. Rapp and D. Andor (2016). Real-time loop closure in 2d lidar slam. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1271–1278.

H. Hirschmuller (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 328–341.

Høglund, Sondre (2014). Autonomous Inspection of Wind Turbines and Buildings using an UAV. (June).

Hornung, Armin and Wurm, Kai M and Bennewitz, Maren and Stachniss, Cyrill and Burgard, Wolfram (2013). OctoMap : an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, *34*(3), 189–206.

V. Indelman and S. Williams and M. Kaess and F. Dellaert (2012). Factor graph based incremental smoothing in inertial navigation systems. *2012 15th International Conference on Information Fusion*. 2154–2161.

International Telecommunication Union (2011). Bt.601 : studio encoding parameters of digital television for standard 4 :3 and wide screen 16 :9 aspect ratios. `http://www.itu.int/rec/R-REC-BT.601/`. (Accessed on 11/19/2017).

Itseez (2017). Open source computer vision library. `https://github.com/itseez/opencv`.

Jayaraman, S (2009). *Digital image processing*. Tata McGraw Hill Education, New Delhi.

T. Kanade and A. Yoshida and K. Oda and H. Kano and M. Tanaka (1996). A stereo machine for video-rate dense depth mapping and its new applications. *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 196–202.

J. Kannala and S. S. Brandt (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(8), 1335–1340.

Kendall, Alex and Martirosyan, Hayk and Dasgupta, Saumitro and Henry, Peter and Kennedy, Ryan and Bachrach, Abraham and Bry, Adam (2017). End-to-end learning of geometry and context for deep stereo regression. *The IEEE International Conference on Computer Vision (ICCV)*.

Keselman, Leonid and Iselin Woodfill, John and Grunnet-Jepsen, Anders and Bhowmik, Achintya (2017). Intel realsense stereoscopic depth cameras. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Khatib, Oussama (1990). *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, Springer New York, New York, NY. 396–404.

Khoshelham, Kourosh and Elberink, Sander Oude (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, *12*(2), 1437–1454.

Ayoung Kim and Ryan M. Eustice (2015). Active visual slam for robotic area coverage : Theory and experiment. *The International Journal of Robotics Research*, *34*(4-5), 457–475.

A. Klaus and M. Sormann and K. Karner (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *18th International Conference on Pattern Recognition (ICPR'06)*. vol. 3, 15–18.

K. Konolige (2010). Projected texture stereo. *2010 IEEE International Conference on Robotics and Automation*. 148–155.

M. Labbé and F. Michaud (2014). Online global loop closure detection for large-scale multi-session graph-based slam. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2661–2666.

R. S. Lim and H. M. La and W. Sheng (2014). A robotic crack inspection and mapping system for bridge deck maintenance. *IEEE Transactions on Automation Science and Engineering*, *11*(2), 367–378.

Lin, Jacob J. and Han, Kevin K. and Golparvar-Fard, Mani (2015). A framework for model-driven acquisition and analytics of visual data using uavs for automated construction progress monitoring. *Computing in Civil Engineering 2015*. 156–164.

Lucas, Bruce D. and Kanade, Takeo (1981). An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'81, 674–679.

S. Lynen and M. W. Achtelik and S. Weiss and M. Chli and R. Siegwart (2013). A robust and modular multi-sensor fusion approach applied to mav navigation. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3923–3929.

E. Marder-Eppstein and E. Berger and T. Foote and B. Gerkey and K. Konolige (2010). The office marathon : Robust navigation in an indoor office environment. *2010 IEEE International Conference on Robotics and Automation*. 300–307.

Matas, J. and Galambos, C. and Kittler, J. (2000). Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*, *78*(1), 119–137.

Mei, Christopher and Sibley, Gabe and Cummins, Mark and Newman, Paul and Reid, Ian (2011). RSLAM : A System for Large-Scale Mapping in Constant-Time Using Stereo. *International Journal of Computer Vision*, *94*(2), 198–214.

Meier, Lorenz and Honegger, Dominik and Vilhjalmsson, Vilhjalmur and Pollefeys, Marc (2017). Real-time stereo matching failure prediction and resolution using orthogonal stereo setups. *2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE*.

M. Montemerlo and S. Thrun and D. Koller and B. Wegbreit (2003). FastSLAM 2.0 : An improved particle filtering algorithm for simultaneous localization and mapping that

provably converges. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Acapulco, Mexico.

T. Moore and D. Stouch (2014). A generalized extended kalman filter implementation for the robot operating system. *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer.

R. Mur-Artal and J. D. Tardós (2017a). Orb-slam2 : An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, *33*(5), 1255–1262.

R. Mur-Artal and J. D. Tardós (2017b). Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, *2*(2), 796–803.

Navigant Research (2015). Revenue for wind turbine unmanned aerial vehicles sales is expected to total nearly 6 billion by 2024 navigant research. `https://www.navigantresearch.com/research/drones-for-wind-turbine-inspection`. (Accédé le 08/17/2017).

Noureldin, Aboelmagd and Karamat, Tashfeen B and Georgy, Jacques (2013). *Introduction*, Springer Berlin Heidelberg, Berlin, Heidelberg.

H. Oleynikova and M. Burri and Z. Taylor and J. Nieto and R. Siegwart and E. Galceran (2016). Continuous-time trajectory optimization for online uav replanning. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 5332–5339.

Oleynikova, Helen and Taylor, Zachary and Fehr, Marius and Siegwart, Roland and Nieto, Juan (2017). Voxblox : Incremental 3d euclidean signed distance fields for on-board mav planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Omar, Tarek and Nehdi, Moncef L. (2018). *Automated Data Collection for Progress Tracking Purposes : A Review of Related Techniques*, Springer International Publishing, Cham. 391–405.

Papachristos, Christos and Khattak, Shehryar and Alexis, Kostas (2017). Uncertainty-aware receding horizon exploration and mapping using aerial robots. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4568–4575.

M. Srinivasan Ramanagopal and A. P. V. Nguyen and J. Le Ny (2017). A motion planning strategy for the active vision-based mapping of ground-level structures. *IEEE Transactions on Automation Science and Engineering*, *PP*(99), 1–13.

Ratliff, Nathan and Zucker, Matt and Bagnell, J Andrew and Srinivasa, Siddhartha (2009). Chomp : Gradient optimization techniques for efficient motion planning. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 489–494.

Rublee, Ethan and Rabaud, Vincent and Konolige, Kurt and Bradski, Gary (2011). Orb : An efficient alternative to sift or surf. *Proceedings of the 2011 International Conference on Computer Vision*. IEEE Computer Society, Washington, DC, USA, ICCV '11, 2564–2571.

S. Rusinkiewicz and M. Levoy (2001). Efficient variants of the icp algorithm. *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 145–152.

Radu Bogdan Rusu (2009). *Semantic 3D object maps for everyday manipulation in human living environments*. Thèse de doctorat, Technical University Munich, Germany.

Radu Bogdan Rusu and Steve Cousins (2011). 3D is here : Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

Schling, Boris (2011). *The Boost C++ Libraries*. XML Press.

W. Sheng and H. Chen and N. Xi (2008). Navigating a miniature crawler robot for engineered structure inspection. *IEEE Transactions on Automation Science and Engineering*, *5*(2), 368–373.

Cyrill Stachniss and Giorgio Grisetti and Wolfram Burgard (2005). Information gain-based exploration using rao-blackwellized particle filters. *In RSS*. 65–72.

Stenning, Braden E. and McManus, Colin and Barfoot, Timothy D. (2013). Planning using a network of reusable paths : A physical embodiment of a rapidly exploring random tree. *Journal of Field Robotics*, *30*(6), 916–950.

Stokkeland, Martin and Klausen, Kristian and Johansen, Tor A. (2015). Autonomous visual navigation of Unmanned Aerial Vehicle for wind turbine inspection. *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, 998–1007.

Sugandi, Budi and Kim, Hyoungseop and Tan, Joo Kooi and Ishikawa, Seiji (2009). A moving object tracking based on color information employing a particle filter algorithm. *Artificial Life and Robotics*, *14*(1), 39–42.

Szeliski, Richard (2011). *Computer vision : algorithms and applications*. Springer.

Thrun, Sebastian and Montemerlo, Mike and Dahlkamp, Hendrik and Stavens, David and Aron, Andrei and Diebel, James and Fong, Philip and Gale, John and Halpenny, Morgan and Hoffmann, Gabriel and others (2006). Stanley : The robot that won the darpa grand challenge. *Journal of field Robotics*, *23*(9), 661–692.

Transports Canada (2016). Exemption de l'application des articles 602.41 et 603.66 du règlement de l'aviation canadien.

Yelda Turkan and Frederic Bosche and Carl T. Haas and Ralph Haas (2012). Automated progress tracking using 4d schedule and 3d sensing technologies. *Automation in Construction*, *22*(Supplement C), 414 – 421. Planning Future Cities-Selected papers from the 2010 eCAADe Conference.

Wang, Chaoqun and Meng, Lili and Li, Teng and De Silva, Clarence W. and Meng, Max Q.-H. (2017). Towards autonomous exploration with information potential field in 3D envi-

ronments. *2017 18th International Conference on Advanced Robotics (ICAR)*. IEEE, 340–345.

Wirth, Stephan and Pellenz, Johannes (2007). Exploration Transform : A stable exploring algorithm for robots in rescue environments. *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*. IEEE, Rome, 1–5.

Xie, Saining and Tu, Zhuowen (2015). Holistically-nested edge detection. *Proceedings of IEEE International Conference on Computer Vision*.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*. IEEE Comput. Soc. Press, 146–151.

Z. Yang and F. Gao and S. Shen (2017). Real-time monocular dense mapping on aerial robots using visual-inertial fusion. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 4552–4559.

Yoder, Luke and Scherer, Sebastian (2016). *Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle*, Springer International Publishing, Cham. 427–440.

Zhang, Huiyi and Jackman, John (2014). Feasibility of Automatic Detection of Surface Cracks in Wind Turbine Blades. *Wind Engineering*, *38*(6), 575–586.

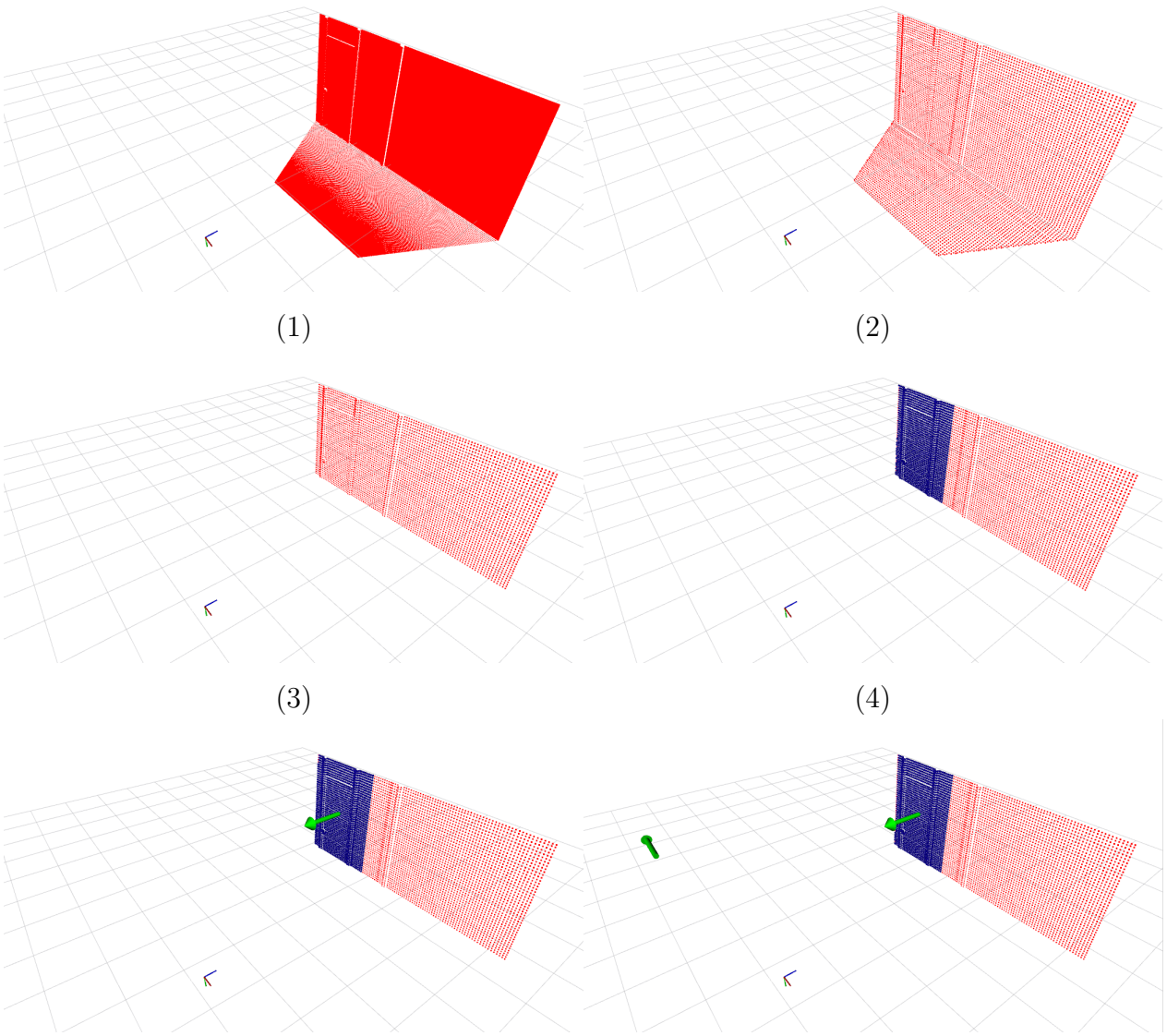
Zhang, Ji and Singh, Sanjiv (2017). Low-drift and real-time lidar odometry and mapping. *Auton. Robots*, *41*(2), 401–416.

Zhang, Ji and Singh, Sanjiv (2018). Aerial and ground-based collaborative mapping : An experimental study. *Field and Service Robotics*. Springer, 397–412.

Z. Zhang (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1330–1334.

Z. Zhang (2012). Microsoft kinect sensor and its effect. *IEEE MultiMedia*, *19*(2), 4–10.

ANNEXE A VISUALISATION DES ÉTAPES DE L'ALGORITHME DE SUIVI DE MUR



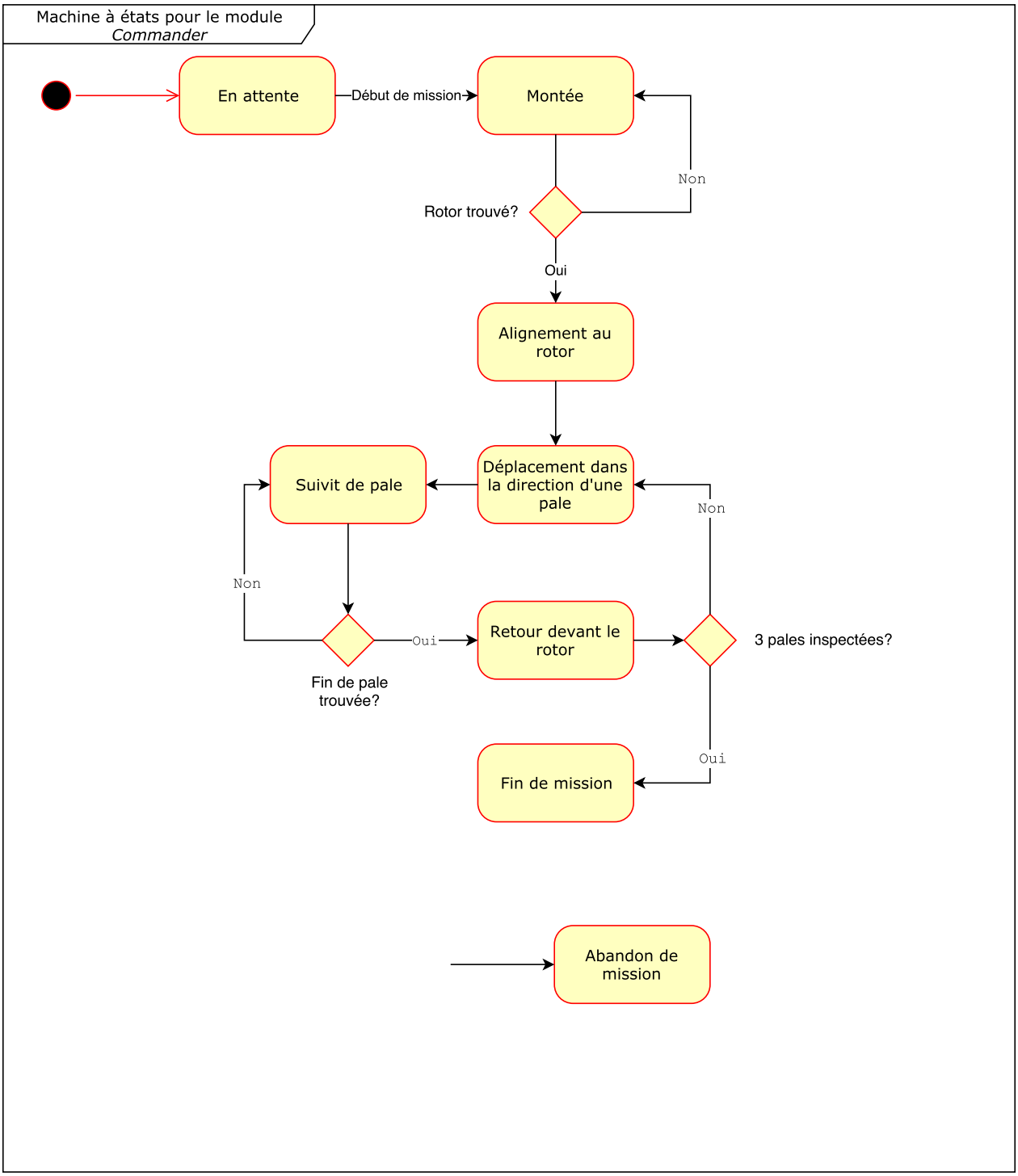
(1) (2)

(3) (4)

(5) (6)

(1) Nuage de points initial. (2) Downsampling du nuage. (3) Filtrage du sol par seuillage de la coordonnée z. (4) Sélection de un tiers du nuage vers l'avant du robot. (5) Calcul de la normale par ACP. (6) Calcul de la nouvelle pose objectif en prenant une longueur de pas dans la direction obtenue par le produit vectoriel entre la normale et l'axe z du robot.

ANNEXE B MACHINE À ÉTATS POUR L'INSPECTION D'ÉOLIENNES



ANNEXE C LISTE DES PUBLICATIONS

[Journal] A. Borowczyk, D.-T. Nguyen, **A. Phu-Van Nguyen**, D. Q. Nguyen, D. Saussié and J. Le Ny, *Autonomous Landing of a Quadcopter on a High-Speed Ground Vehicle*. AIAA Journal of Guidance, Control and Dynamics, In Press, 2017.

[Conférence] A. Borowczyk, D.-T. Nguyen, **A. Phu-Van Nguyen**, D. Q. Nguyen, D. Saussié and J. Le Ny, *Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle*. Proceedings of the IFAC World Congress, Toulouse, France, July 2017.

[Journal] M. S. Ramanagopal, **A. Phu-Van Nguyen** and J. Le Ny, A Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures. Accepted by Transactions on Automation Science and Engineering, November 2017.

Le travail des deux premières publications à propos de l'atterrissage d'un quadricoptère sur une voiture en mouvement a été réalisé dans le cadre de la participation du *Mobile Robotics and Autonomous Systems Laboratory* au DJI Challenge lors des session d'hiver et d'été 2016. N'étant pas directement liées au présent mémoire, elles sont mentionnées ici car plusieurs des méthodes de contrôle de véhicule aérien et de traitement d'images ont été apprises lors de notre participation à cette compétition. Mon rôle spéficique dans ce projet était l'aide au traitement d'images pour la détection de la plateforme d'atterrissage et l'aide au niveau de l'implémentation du système en C++.

La dernière publication intitulée *A Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures* a été initialement soumise en février 2016 par M. S. Ramanagopal et J. Le Ny au journal *Transactions on Automation Science and Engineering* (T-ASE). Lors du retour de l'évaluation par les pairs, il a été demandé entres autres d'ajouter un volet expérimental à l'article afin de prouver la validité des algorithmes développés dans l'article. C'est à ce moment, lors de la session d'automne 2016 et au début de la session d'hiver 2017 que je me suis ajouté au projet pour faire l'implémentation sur un robot physique. Au moment de l'écriture du présent mémoire, l'article a été accepté pour publication dans une édition future du journal T-ASE. Nous présentons le travail réalisé dans le cadre de ce projet au Chapitre 3.